10/018836
JC13 Rec'd PCT/PTO 19 DEC 2001

101 | prtb

DESCRIPTION

Encoding Apparatus and Method, Recording Medium and Program

Technical Field

This invention relates to an encoding method and apparatus, a recording medium and a program. More particularly, it relates to an encoding method and apparatus, a recording medium and a program in which the management information of contents of data recorded on the recording medium is rendered into a file for recording.

Background Art

Recently, a variety of types of optical discs have been proposed as a recording medium that can be removed from a recording apparatus. These recordable optical discs have been proposed as a large capacity medium of several GBs and are thought to be promising as a medium for recording AV (audio visual) signals, such as video signals. Among the digital AV signal sources (supply sources), to be recorded on this recordable optical disc, there are CS digital satellite broadcast and BS digital broadcast. Additionally, the ground wave television broadcast of the digital system has also been proposed for future use.

The digital video signals, supplied from these sources, are routinely image-compressed under the MPEG (Moving Picture Experts Group) 2 system. In a

recording apparatus, a recording rate proper to the apparatus is set. If digital video signals of the digital broadcast are recorded in the conventional image storage mediums for domestic use, digital video signals are first decoded and subsequently bandwidth-limited for recording. In the case of the digital recording system, including, of course, the MPEG1 Video, MPEG2 video and DV system, digital video signals are first decoded and subsequently re-encoded in accordance with an encoding system for the recording rate proper to the apparatus for subsequent recording.

However, this recording system, in which the supplied bitstream is decoded once and subsequently bandwidth-limited and re-encoded prior to recording, suffers from deteriorated picture quality. If, in recording image-compressed digital signals, the transmission rate of input digital signals is less than the recording rate for the recording and/or reproducing apparatus, the method of directly recording the supplied bitstream without decoding or re-encoding suffers from deterioration in the picture quality only to the least extent. However, if the transmission rate of the input digital signals exceeds the recording rate of the recording and/or reproducing apparatus, it is indeed necessary to re-encode the bitstream and to record the re-encoded bitstream, so that, after decoding in the recording and/or reproducing apparatus, the transmission rate will be not higher than the upper limit of the disc recording rate.

If the bitstream is transmitted in a variable rate system in which the bit rate of the input digital signal is increased or decreased with time, the capacity of the recording medium can be exploited less wastefully with a disc recording apparatus

adapted for transiently storing data in a buffer and for recording the data in a burst fashion than with a tape recording system having a fixed recording rate imposed by the fixed rpm of the rotary head.

Thus, it may be predicted that, in the near future when the digital broadcast is to become the mainstream, an increasing demand will be raised for a recording and/or reproducing apparatus in which broadcast signals are recorded as digital signals, without decoding or re-encoding, as in a DataStreamer, and in which a disc is used as a recording medium.

If the recording medium is increased in capacity, a larger volume of data (herein the image or speech pertinent to a program) can be recorded on the recording medium. So, a larger number of programs are recorded on one disc, with the result that the operation of the user selecting a desired one of many programs recorded in the disc is complex. Thus, a necessity is felt for enabling a user to confirm data recorded in reproducing the disc to enable a desired program (data) to be selected extremely readily.

Disclosure of the Invention

It is therefore an object of the present invention to render the management information for contents of data recorded on a recording medium for recording whereby the contents of data recorded on the recording medium and the replay information can be managed properly.

In one aspect, the present invention provides an encoding apparatus for encoding picture data, including an encoder for encoding the picture data at a variable rate, and a controller for managing control so that the amount of picture coding data will be substantially proportionate to the time lapse.

The controller may manage control so that stuffing bytes will be encoded if the amount of the picture coding data generated per unit time is less than a preset value.

The controller may verify whether or not the stuffing bytes will be encoded depending on the amount of data generated in encoding respective pictures.

The controller may manage control of encoding the stuffing bytes so that no overflow will be produced in the VBV buffer.

The controller may manages control to perform encoding in an encoding mode in which the amount of the picture coding data is substantially proportionate to the lapse of time or in a routine encoding mode.

The controller may generate the additional information indicating whether or not the encoding mode is such encoding mode in which the amount of the picture coding data is substantially proportionate to the lapse of time.

In another aspect, the present invention provides a encoding method for encoding picture data, including an encoding step of encoding the picture data at a variable rate, and a controlling step of managing control so that the amount of picture coding data will be substantially proportionate to the time lapse.

In still another aspect, the present invention provides a recording medium

having recorded thereon a computer-readable program used for controlling an encoding apparatus encoding picture data; the program including an encoding step of encoding the picture data at a variable rate and a controlling step of managing control so that the amount of picture coding data will be substantially proportionate to the time lapse.

In still another aspect, the present invention provides a program for causing a computer controlling an encoding apparatus encoding picture data to execute an encoding step of encoding the picture data at a variable rate, and a controlling step of managing control so that the amount of picture coding data will be substantially proportionate to the time elapsed.

In the encoding method and apparatus, recording medium and the program according to the present invention, picture data are encoded at a variable rate, and control is managed so that the amount of encoding picture data will be approximately proportionate to time elapsed.

In yet another aspect, the present invention provides a recording medium having picture data recorded thereon, the recording medium having recorded thereon an AV stream file including the picture data and audio data associated with the picture data, and a flag indicating the recording mode of the AV stream file.

The flag may be time_controlled_flag.

The flag may indicate that the mode is such a mode in which the recording is made such that the file size will be proportionate to the time elapsed as from the time

of recording.

In the recording medium according to the present invention, there are recorded picture data, an AV stream file containing audio data associated with the picture data and a flag indicating the recording mode of the AV stream file.

Other objects, features and advantages of the present invention will become more apparent from reading the embodiments of the present invention as shown in the drawings.

Brief Description of the Drawings

Fig.1 shows a configuration of an embodiment of a recording and/or reproducing apparatus according to the present invention.

Fig.2 illustrates the data format of data recorded on a recording medium by a recording and/or reproducing apparatus 1.

Fig.3 illustrates Real PlayList and Virtual PlayList.

Figs.4A, 4B and 4C illustrate the creation of the Real PlayList.

Figs.5A, 5B and 5C illustrate deletion of the Real PlayList.

Figs.6A and 6B illustrate assemble editing.

Fig.7 illustrates provision of a sub path in the Virtual PlayList.

Fig.8 illustrates the changing of the playback sequence of the PlayList.

Fig.9 illustrates a mark on the PlayList and a mark on the Clip.

Fig.10 illustrates a menu thumbnail.

Fig.11 illustrates mark added to the PlayList.

Fig.12 illustrates a mark added to the Clip.

Fig.13 illustrates the relation between the PlayList, Clip and the thumbnail file.

Fig.14 illustrates a directory structure.

Fig.15 illustrates a syntax of infr.dvr.

Fig.16 shows a syntax of DVRVolume.

Fig.17 shows a syntax of ResumeVolume.

Fig.18 shows a syntax of UIAppInfoVolume.

Fig.19 shows a table of character set values.

Fig.20 shows a syntax of TableOfPlayList.

Fig.21 shows another syntax of TableOfPlayList.

Fig.22 shows a syntax of the MakersPrivateData.

Fig.23 shows a syntax of xxxx.rpls and yyyy.vpls.

Figs.24A to 24C illustrate the PlayList.

Fig.25 shows a syntax of PlayList.

Fig.26 shows a table of PlayList_type.

Fig.27 shows a syntax of UIAppInfoPlayList.

Figs.28A to 28C illustrate flags in the UIAppInfoPlayList syntax shown in Fig.27.

Fig.29 illustrates a PlayItem.

Fig.30 illustrates a PlayItem.

Fig.31 illustrates a PlayItem.

Fig.32 shows a syntax of the PlayItem.

Fig.33 illustrates IN-time.

Fig.34 illustrates OUT-time.

Fig.35 shows a table of Connection_Condition.

Figs.36A to 36D illustrate Connection_Condition.

Fig.37 illustrates BridgeSequenceInfo.

Fig.38 shows a syntax of BridgeSequenceInfo.

Fig.39 illustrates SubPlayItem.

Fig.40 shows a syntax of SubPlayItem.

Fig.41 shows a table of Mark_type.

Fig.42 shows a syntax of PlayListMark.

Fig.43 shows a table of Mark_type.

Fig.44 illustrates Mark_time_stamp.

Fig.45 shows a syntax of zzzzz.clip.

Fig.46 shows a syntax of ClipInfo.

Fig.47 shows a table of Clip_stream_type.

Fig.48 illustrates offset_SPN.

Fig.49 illustrates offset_SPN.

Figs.50A, 50B illustrate the STC domain.

Fig.51 illustrates STC_Info.

Fig.52 shows a syntax of STC_Info.

Fig.53 illustrates ProgramInfo.

Fig.54 shows a syntax of ProgramInfo.

Fig.55 shows a syntax of VideoCondingInfo.

Fig.56 shows a table of Video_format.

Fig.57 shows a table of frame_rate.

Fig.58 shows a table of display_aspect_ratio.

Fig.59 shows a syntax of AudioCondingInfo.

Fig.60 shows a table of audio_coding.

Fig.61 shows a table of audio_component_type.

Fig.62 shows a table of sampling_frequency.

Fig.63 illustrates CPI.

Fig.64 illustrates CPI.

Fig.65 shows a syntax of CPI.

Fig.66 shows a table of CPI_type.

Fig.67 illustrates video EP_map.

Fig.68 illustrates EP_map.

Fig.69 illustrates EP_map.

Fig.70 shows a syntax of EP_map.

Fig.71 shows a table of EP_typevalues.

Fig.72 shows a syntax of EP_map_for_one_stream_PID.

Fig.73 illustrates TU_map.

Fig.74 shows a syntax of TU_map.

Fig.75 shows a syntax of ClipMark.

Fig.76 shows a table of Mark_type.

Fig.77 shows a table of Mark_type_stamp.

Fig.78 shows a syntax of menu.thmb and mark.thmb.

Fig.79 shows the syntax of thumbnail.

Fig.80 shows a table of thumbnail_picture_format.

Figs.81A and 81Billustrate tn_block.

Fig.82 illustrates a structure of a transport stream of DVR MPEG2.

Fig.83 shows a recorder model of a transport stream of DVR MPEG2.

Fig.84 shows a player model of a transport stream of DVR MPEG2.

Fig.85 shows the syntax of a source packet.

Fig.86 shows the syntax of TP_extra_header.

Fig.87 shows a table of a copy permission indicator.

Fig.88 illustrates seamless connection.

Fig.89 illustrates seamless connection.

Fig.90 illustrates seamless connection.

Fig.91 illustrates seamless connection.

Fig.92 illustrates seamless connection.

Fig.93 illustrates audio overlap.

Fig.94 illustrates seamless connection employing BridgeSequence.

Fig.95 illustrates seamless connection not employing BridgeSequence.

Fig.96 shows a DVR STD model.

Fig.97 is a timing chart for decoding and display.

Fig.98 illustrates the operation of the AV encoder of Fig.1.

Fig.99 is a flowchart for illustrating the operation of encoding video with a variable bitrate to record an AV stream.

Fig.100 illustrates Video Buffering Verifier.

Fig.101 illustrates VBV control.

Fig.102 illustrates VBV control.

Fig.103 shows a case of controlling the variable bit rate.

Fig.104 shows a case of control of the variable bit rate.

Fig.105 is a flowchart for illustrating details of step S21 of Fig.99.

Fig.106 is a flowchart for illustrating details of step S205 of Fig.106.

Fig.107 illustrates the relation between time lapse of an AV stream and the amount of the data bytes on the AV stream.

Fig.108 is a flowchart for illustrating the operation of encoding video with a variable bit rate and of recording an AV stream.

Fig.109 is a flowchart for illustrating details of step S400 of Fig.108.

Fig.110 is a flowchart for illustrating the encoding mode of guaranteeing the proportional relation between the time lapse the amount of data bytes of the AV

stream.

Fig.111 shows an instance of the minimizing operation.

Fig.112 shows an instance of erasing unneeded stream data before IN_time in case of minimizing.

Fig.113 shows an instance of erasing unneeded stream data at back of OUT_time in case of maximizing.

Fig.114 is a flowchart for illustrating a typical operation of forming EP_map.

Fig.115 illustrates a medium.

Best Mode for Carrying out the Invention

Referring to the drawings, present embodiment of the present invention will be explained in detail. Fig.1 shows a typical inner structure of a recording and/or reproducing apparatus 1 embodying the present invention. First, the structure of a recording unit 2, configured for recording signals input from outside, is explained. The recording and/or reproducing apparatus 1 is configured for being fed with and recording analog or digital data.

Analog video signals and analog audio signals are fed to terminals 11, 12, respectively. The video signals, input to the terminal 11, are output to an analysis unit 14 and to an AV encoder 15. The audio signals, input to the terminal 12, are output to the analysis unit 14 and to the AV encoder 15. The analysis unit 14 extracts feature points, such as scene changes, from the input video and audio signals.

The AV encoder 15 encodes input video and audio signal to output the system information (S), such as an encoded video stream (V), an encoded audio stream (A) and AV synchronization, to a multiplexer 16.

The encoded video stream is a video stream encoded e.g., with the MPEG (Moving Picture Expert Group) 2 system, whilst the encoded audio stream is an audio stream encoded in accordance with the MPEG1 system, with the encoded audio stream being e.g., an audio stream encoded in e.g., the MPEG1 system or an audio stream encoded in accordance with the Dolby AC3 (trademak) system. The multiplexer 16 multiplexes the input video and audio streams, based on the input system information, to output a multiplexed stream through a switch 17 to a multiplexed stream analysis unit 18 and to a source packetizer 19.

The multiplexed stream is e.g., an MPEG-2 transport stream or an MPEG2 program stream. The source packetizer 19 encodes the input multiplexed stream into an AV stream composed of source packets in accordance with an application format of a recording medium 100 on which to record the stream. The AV stream is processed in ECC (error correction and coding) unit 20 and a modulation unit 21 with appendage of ECC codes and with modulation, before being output to a write unit 22, which then writes (records) an AV stream file based on a control signals output by the controller 23.

The transport stream, such as digital television broadcast, input from a digital interface or a digital television tuner, is input to a terminal 13. There are two

recording systems for recording the transport stream input to the terminal 13, one being a transparent recording system and the other being a system in which recording is preceded by re-encoding aimed to lower e.g., the recording bit rate. The recording system command information is input from a terminal 24 as a user interface to a controller 23.

In the transparent recording of the input transport stream, a transport stream, input to a terminal 13, is output through a switch 17 to a multiplexed stream analysis unit 18 and to the source packetizer 19. The ensuing processing of recording an AV stream on a recording medium is the same as that of encoding and recording analog input audio and video signals, as described above, and hence is not explained here for simplicity.

If the input transport stream is re-encoded and subsequently recorded, the transport stream, input to the terminal 13, is fed to a demultiplexer 26, which demultiplexes the input transport stream to extract a video stream (V), an audio stream (A) and the system information (S).

Of the stream (information), as extracted by the demultiplexer 26, the video stream is output to an audio decoder 27, whilst the audio stream and the system information are output to the multiplexer 16. The audio decoder 27 decodes the input transport stream to output the encoded video stream (V) to the multiplexer 16.

The audio stream and the system information, output from the demultiplexer 26 and input to the multiplexer 16, and the video stream, output by the AV encoder 15,

are multiplexed, based on the input system information, and output to the multiplexed stream analysis unit 18 and to the source packetizer 19 through switch 17, as a multiplexed stream. The ensuing processing of recording an AV stream on a recording medium is the same as that of encoding and recording analog input audio and video signals, as described above, and hence is not explained here for simplicity.

The recording and/or reproducing apparatus 1 of the present embodiment records a file of the AV stream on the recording medium 100, while also recording the application database information which accounts for the file. The input information to the controller 23 is the feature information for the moving picture from the analysis unit 14, the feature information of the AV stream from the multiplexed stream analysis unit 18 and the user command information input at a terminal 24.

The feature information of the moving picture, supplied from the analysis unit 14, is generated by the analysis unit 14 when the AV encoder 15 encodes video signals. The analysis unit 14 analyzes the contents of the input video and audio signals to generate the information pertinent to the pictures characteristic of the input moving picture signals (clip mark). This information is the information indicating a picture of characteristic clip mark points, such as program start points, scene change points, CM commercial start and end points, title or telop in input video signals, and also includes a thumbnail of the picture and the information pertinent to stereo/monaural switching points and muted portions of audio signals.

The above picture indicating information is fed through controller 23 to the

multiplexer 16. When multiplexing a encoded picture specified as clip mark by the controller 23, the multiplexer 16 returns the information for specifying the encoded picture on the AV stream to the controller 23. Specifically, this information is the PTS (presentation time stamp) of a picture or the address information on the AV stream of an encoded version of the picture. The controller 23 stores the sort of feature pictures and the information for specifying the encoded picture on the AV stream in association with each other.

The feature information of the AV stream from the multiplexed stream analysis unit 18 is the information pertinent to the encoding information of the AV stream to be recorded, and is recorded by an analysis unit 18. For example, the feature information includes the time stamp and address information of the I-picture in the AV stream, discontinuous point information of system time clocks, encoding parameters of the AV stream and change point information of the encoding parameters in the AV stream. When transparently recording the transport stream, input from the terminal 13, the multiplexed stream analysis unit 18 detects the picture of the aforementioned clip mark, from the input transport stream, and generates the information for specifying a picture designated by the clip mark and its type.

The user designation information from the terminal 24 is the information specifying the playback domain, designated by the user, character letters for explaining the contents of the playback domain, or the information such as bookmarks or resuming points set by the user for his or her favorite scene.

Based on the aforementioned input information, the controller 23 creates a database of the AV stream (Clip), a database of a group (PlayList) of playback domains (PlayItem) of the AV stream, management information of the recorded contents of the recording medium 100 (info.dvr) and the information on thumbnail pictures. Similarly to the AV stream, the application database information, constructed from the above information, is processed in the ECC unit 20 and the modulation unit 21 and input to the write unit 22, which then records a database file on the recording medium 100.

The above-described application database information will be explained subsequently in detail.

When the AV stream file recorded on the recording medium 100 (files of picture data and speech data) and the application database information, thus recorded on the recording medium 100, are reproduced by a reproducing unit 3, the controller 23 first commands a readout unit 28 to read out the application database information from the recording medium 100. The readout unit 28 reads out the application database information from the recording medium 100, which then reads out the application database information from the recording medium 100 to send the application database information through demodulation and error correction processing by a demodulating unit 29 and an ECC decoder 30 to the controller 23.

Based on the application database information, the controller 23 outputs a list of PlayList recorded on the recording medium 100 to a user interface of the terminal

24. The user selects the PlayList, desired to be reproduced, from the list of PlayLists. The information pertinent to PlayList, specified to be reproduced, is input to the controller 23. The controller 23 commands the readout unit 28 to read out the AV stream file necessary in reproducing the PlayList. In accordance with the command, the readout unit 28 reads out the corresponding AV stream from the recording medium 100 to output the read-out AV stream to the demodulating unit 29. The AV stream, thus input to the demodulating unit 29, is demodulated by preset processing and output through the processing by the ECC decoder 30 to a source depacketizer 31.

The source depacketizer 31 converts the AV stream of the application format, read out from the recording medium 100 and processed in a preset fashion, into a stream processable by the demultiplexer 26. The demultiplexer 26 outputs the system information (S), such as the video stream (V), audio stream (A) or the AV synchronization, forming the playback domain (PlayItem) of the AV stream specified by the controller 23, to the audio decoder 27, which AV decoder 27 decodes the video stream and the audio stream to output the playback video signal and the playback audio signal to associated terminals 32, 33, respectively.

If fed from the terminal 24, as a user interface, with the information instructing random access playback or special playback, the controller 23 determines the readout position of the AV stream from the recording medium 100, based on the contents of the database (Clip) of the AV stream, to command the readout unit 28 to read out the AV stream. If the PlayList as selected by the user is to be reproduced as from a preset

time point, the controller 23 commands the readout unit 28 to read out data from an I-picture having a time stamp closest to the specified time point.

When the user has selected a certain clip mark from indexing points or scene change points for the program stored in the ClipMark in the Clip Information, as when the user selects a certain picture from a list of thumbnail pictures, as demonstrated on a user interface, of the indexing points or scene change points stored in the ClipMark, the controller 23 determines the AV stream readout position from the recording medium 100 to command the readout unit 28 to read out the AV stream. That is, the controller 23 commands the readout unit 28 to read out data from an I-picture having an address closest to the address on the AV stream which has stored the picture selected by the user. The readout unit 28 reads out data from the specified address . The read-out data is processed by the demodulating unit 29, ECC decoder 30 and by the source packetizer 19 so as to be supplied to the demultiplexer 26 and decoded by the audio decoder 27 to reproduce AV data indicated by an address of the mark point picture.

If the user has commanded fast forward playback, the controller 23 commands the readout unit 28 to sequentially read out I-picture data in the AV stream in succession based on the database (Clip) of the AV stream.

The readout unit 28 reads out data of the AV stream from a specified random access point. The so read-out data is reproduced through processing by various components on the downstream side.

The case in which the user edits the AV stream recorded on the recording medium 100 is now explained. If desired to specify a playback domain for the AV stream recorded on the recording medium 100, for example, if desired to create a playback route of reproducing a portion sung by a singer A from a song program A, and subsequently reproducing a portion sung by the same singer A from another song program B, the information pertinent to a beginning point (IN-point) and an end point (OUT-point) of the playback domain is input to the controller 23 from the terminal as a user interface. The controller 23 creates a database of the group (PlayList) of playback domains (PlayItem) of the AV streams.

When the user desires to erase a portion of the AV stream recorded on the recording medium 100, the information pertinent to the IN-point and the OUT-point of the erasure domain is input to the controller 23, which then modifies the database of the PlayList so as to refer to only the needed AV streams. The controller 23 also commands the write unit 22 to erase an unneeded stream portion of the AV stream.

The case in which the user desires to specify playback domains of an AV stream recorded on the recording medium to create a new playback route, and to interconnect the respective playback domains in a seamless fashion, is now explained. In such case, the controller 23 creates a database of a group (PlayList) of the playback domains (PlayItem) of the AV stream and undertakes to partially re-encode and re-multiplex the video stream in the vicinity of junction points of the playback domains.

The picture information at the IN-point and that at the OUT-point of a playback

domain are input from a terminal 24 to a controller 23. The controller 23 commands the readout unit 28 to read out data needed to reproduce the pictures at the IN-point and at the OUT-point. The readout unit 28 reads out data from the recording medium 100. The data so read out is output through the demodulating unit 29, ECC decoder 30 and the source packetizer 19 to the demultiplexer 26.

The controller 23 analyzes data input to the demultiplexer 26 to determine the re-encoding method for the video stream (change of picture_coding_type and assignment of the quantity of encoding bits for re-encoding) and the re-multiplexing system to send the system to the AV encoder 15 and to the multiplexer 16.

The demultiplexer 26 then separates the input stream into the video stream (V), audio stream (A) and the system information (S). The video stream may be classed into data input to the audio decoder 27 and data input to the multiplexer 16. The former is data needed for re-encoding, and is decoded by the audio decoder 27, with the decoded picture being then re-encoded by the AV encoder 15 and thereby caused to become a video stream. The latter data is data copied from an original stream without re-encoding. The audio stream and the system information are directly input to the multiplexer 16.

The multiplexer 16 multiplexes an input stream, based on the information input from the controller 23, to output a multiplexed stream, which is processed by the ECC unit 20 and the modulation unit 21 so as to be sent to the write unit 22. The write unit 22 records an AV stream on the recording medium 100 based on the control signals

supplied from the controller 23.

The application database information and the operation based on this information, such as playback and editing, are hereinafter explained. Fig.2 shows the structure of an application format having two layers, that is PlayList and Clip, for AV stream management. The Volume Information manages all Clips and PlayLists in the disc. Here, one AV stream and the ancillary information thereof, paired together, is deemed to be an object, and is termed Clip. The AV stream file is termed a Clip AV stream file, with the ancillary information being termed the Clip Information file.

One Clip AV stream file stores data corresponding to an MPEG-2 transport stream arranged in a structure prescribed by the application format. By and large, a file is treated as a byte string. The contents of the Clip AV stream file are expanded on the time axis, with entry points in the Clip (I-picture) being mainly specified on the time basis. When a time stamp of an access point to a preset Clip is given, the Clip Information file is useful in finding the address information at which to start data readout in the Clip AV stream file.

Referring to Fig.3, PlayList is now explained, which is provided for a user to select a playback domain desired to be viewed from the Clip and to edit the playback domain readily. One PlayList is a set of playback domains in the Clip. One playback domain in a preset Clip is termed PlayItem and is represented by a pair of an IN-point and an OUT-point on the time axis. So, the PlayList is formed by a set of plural PlayItems.

The PlayList is classified into two types, one of which is Real PlayList and the other of which is Virtual PlayList. The Real PlayList co-owns stream portions of the Clip it is referencing. That is, the Real PlayList takes up in the disc the data capacity corresponding to a stream portion of the Clip it is referencing and, when Real PlayList is erased, the data of the stream portion of the Clip it is referencing is also erased.

The Virtual PlayList is not co-owning Clip data. Therefore, if the Virtual PlayList is changed or erased, the contents of the Clip are in no way changed.

The editing of the Real Playlist is explained. Fig.4A shows creation of Real PlayList and, if the AV stream is recorded as a new Clip, the Real PlayList which references the entire Clip is a newly created operation.

Fig.4B shows the division of the real PlayList, that is the operation of dividing the Real PlayList at a desired point to split the Real PlayList in two Real PlayLists. This division operation is performed when two programs are managed in one clip managed by a sole PlayList and when the user intends to re-register or re-record the programs as separate individual programs. This operation does not lead to alteration of the Clip contents, that is to division of the Clip itself.

Fig.4C shows the combining operation of the Real PlayList which is the operation of combining two Real PlayLists into one new Real PlayList. This combining operation is performed such as when the user desires to re-register two programs as a sole program. This operation does not lead to alteration of the Clip contents, that is to combining the clip itself into one.

Fig.5A shows deletion of the entire Real PlayList. If the operation of erasing the entire preset Real PlayList, the associated stream portion of the Clip referenced by the deleted Real PlayList is also deleted.

Fig.5B shows partial deletion of the Real PlayList. If a desired portion of the Real PlayList is deleted, the associated PlayItem is altered to reference only the needed Clip stream portion. The corresponding stream portion of the Clip is deleted.

Fig.5C shows the minimizing of the Real PlayList. It is an operation of causing the PlayItem associated with the Real PlayList to reference only the stream portion of the Clip needed for Virtual PlayList. The corresponding stream portion of the Clip not needed for the Virtual PlayList is deleted.

If the Real PlayList is changed by the above-described operation such that the stream portion of the Clip referenced by the Real PlayList is deleted, there is a possibility that the Virtual PlayList employing the deleted Clip is present such that problems may be produced in the Virtual PlayList due to the deleted Clip.

In order to prevent this from occurring, such a message which runs: "If there exists the Virtual PlayList referencing the stream portion of the Clip the Real PlayList is referencing, and the Real PlayList is deleted, the Virtual PlayList itself is deleted – is it all right?" is displayed for the user in response to the user's operation of deletion by way of confirmation or alarming, after which the processing for deletion is executed or cancelled subject to user's commands. Alternatively, the minimizing operation for the Real PlayList is performed in place of deleting the Virtual PlayList.

The operation for the Virtual PlayList is now explained. If an operation is performed for the Virtual PlayList, the contents of the Clip are not changed. Figs.6A and 6B show the assembling and editing (IN–OUT editing). It is an operation of creating PlayItem of the playback domain the user has desired to view to create Virtual PlayList. The seamless connection between PlayItems is supported by the application format, as later explained.

If there exist two Real PlayLists 1, 2 and clips 1, 2 associated with the respective Real PlayLists, the user specifies a preset domain in the Real PlayList 1 (domain from IN1 to OUT1: PlayItem 1) as the playback domain, and also specifies, as the domain to be reproduced next, a preset domain in the Real PlayList 2 (domain from IN2 to OUT2: PlayItem 2) as the playback domain, as shown in Fig.6A, a sole Virtual PlayList made up of PlayItem 1 and the PlayItem2 is prepared, as shown in Fig.6B.

The re-editing of the Virtual PlayList is now explained. The re-editing may be enumerated by alteration of IN- or OUT points in the Virtual PlayList, insertion or appendage of new PlayItems to the Virtual PlayList and deletion of PlayItems in the Virtual PlayList. The Virtual PlayList itself may also be deleted.

Fig.7 shows the audio dubbing (post recording) to the Virtual PlayList. It is an operation of registering the audio post recording to the Virtual PlayList as a sub path. This audio post recording is supported by the application software. An additional audio stream is added as a sub path to the AV stream of the main path of the Virtual

PlayList.

Common to the Real PlayList and the Virtual PlayList is an operation of changing (moving) the playback sequence of the PlayList shown in Fig.8. This operation is an alteration of the playback sequence of the PlayList in the disc (volume) and is supported by TableOfPlayList as defined in the application format, as will be explained subsequently with reference to e.g., Fig.20. This operation does not lead to alteration of the Clip contents.

The mark (Mark) is now explained. The mark is provided for specifying a highlight or characteristic time in the Clip and in the PlayList, as shown in Fig.9. The mark added to the Clip is termed the ClipMark. The ClipMark is e.g., a program indexing point or a scene change point for specifying a characteristic scene ascribable to contents in the AV stream. The ClipMark is generated by e.g., the analysis unit 14 of Fig.1. When the PlayList is reproduced, the mark of the Clip referenced by the PlayList may be referenced and used.

The mark appended to the PlayList is termed the PlayListMark (play list mark). The PlayListMark is e.g., a bookmark point or a resuming point as set by the user. The setting of the mark to the Clip and to the PlayList is by adding a time stamp indicating the mark time point to the mark list. On the other hand, mark deletion is removing the time stamp of the mark from the mark list. Consequently, the AV stream is in no way changed by mark setting or by mark deletion.

As another format of the ClipMark, a picture referenced by the ClipMark may

be specified on the address basis in the AV stream. Mark setting on the Clip is by adding the address basis information indicating the picture of the mark point to the mark list. On the other hand, mark deletion is removing the address basis information indicating the mark point picture from the mark list. Consequently, the AV stream is in no way changed by mark setting or by mark deletion.

A thumbnail is now explained. The thumbnail is a still picture added to the Volume, PlayList and Clip. There are two sorts of the thumbnail, one of them being a thumbnail as a representative picture indicating the contents. This is mainly used in a main picture in order for the user to select what the or she desired to view on acting on a cursor, not shown. Another sort of the thumbnail is a picture indicating a scene pointed by the mark.

The Volume and the respective PlayLists need to own representative pictures. The representative pictures of the Volume are presupposed to be used for initially demonstrating a still picture representing the disc contents when the disc is set in position in the recording and/or reproducing apparatus 1. It is noted that the disc means the recording medium 100 which is presupposed to be a of disc shape. The representative picture of the PlayList is presupposed to be used as a still picture for representing PlayList contents.

As the representative picture of the PlayList, it may be contemplated to use the initial picture of the PlayList as the thumbnail (representative picture). However, the leading picture at the playback time of 0 is not necessarily an optimum picture

representing the contents. So, the user is allowed to set an optional picture as a thumbnail of the PlayList. Two sorts of the thumbnails, that is the thumbnail as a representative picture indicating the Volume and the thumbnail as a representative picture indicating PlayList, are termed menu thumbnails. Since the menu thumbnails are demonstrated frequently, these thumbnails need to be read out at an elevated speed from the disc. Thus, it is efficient to store the totality of the menu thumbnails in a sole file. It is unnecessary for the menu thumbnails to be pictures extracted from the moving pictures in the volume, but may be a picture captured from a personal computer or a digital still camera, as shown in Fig.10.

On the other hand, the Clip and the PlayList need be marked with plural marks, whilst the pictures of the mark points need to be readily viewed in order to grasp the contents of the mark positions. The picture indicating such mark point is termed a mark thumbnail. Therefore, the picture which is the original of the mark thumbnail is mainly an extracted mark point picture rather than a picture captured from outside.

Fig.11 shows the relation between the mark affixed to the PlayList and the mark thumbnail, whilst Fig.12 shows the relation between the mark affixed to the Clip and the mark thumbnail. In distinction from the menu thumbnail, the mark thumbnail is used in e.g., a sub-menu for representing details of the PlayList, while it is not requested to be read out in a short access time. So, whenever a thumbnail is required, the recording and/or reproducing apparatus 1 opens a file and reads out a portion of the file, while there is no problem presented even if file opening and reading out a

portion of the file by the recording and/or reproducing apparatus 1 takes some time.

For decreasing the number of files present in a volume, it is preferred for the totality of the mark thumbnails to be stored in one file. While the PlayList may own one menu thumbnail and plural mark thumbnails, the user is not required to select the Clip directly (usually, the Clip is selected through PlayList), and hence there is no necessity of providing menu thumbnails.

Fig.13 shows the relation between the menu thumbnails, mark thumbnails, PlayList and Clips. In the menu thumbnail file are filed menu thumbnails provided from one PlayList to another. In the menu thumbnail file is contained a volume thumbnail representing the contents of data recorded on the disc. In the menu thumbnail file are filed thumbnails created from one PlayList to another and from one Clip to another.

The CPI (Characteristic Point Information) is hereinafter explained. The CPI is data contained in the Clip information file and is used mainly for finding a data address in the Clip AV stream file at which to start the data readout when a time stamp of the access point to the Clip is afforded. In the present embodiment two sorts of the CPI are used, one of them being EP_map and the other being TU_map.

The EP_map is a list of entry point (EP) data extracted from the elementary stream and the transport stream. This has the address information used to find the site of entry points in the AV stream at which to start the decoding. One EP data is made up of a presentation time stamp (PTS) and a data address in the AV stream of the

accessing unit associated with the PTS, with the data address being paired to the PTS.

The EP_map is used mainly for two purposes. First, it is used for finding a data address in the AV stream in the accessing unit referenced by the PTS in the PlayList. Second, the EP_map is used for fast forward playback or fast reverse playback. If, in recording the input AV stream by the recording and/or reproducing apparatus 1, the syntax of the stream can be analyzed, the EP_map is created and recorded on the disc.

The TU_map has a list of time unit (TU) data which is derived from the arrival time point of the transport packet input through a digital interface. This affords the relation between the arrival-time-based time and the data address in the AV stream. When the recording and/or reproducing apparatus 1 records an input AV stream, and the syntax of the stream cannot be analyzed, a TU_map is created and recorded on the disc.

The STCInfo stores the discontinuous point information in the AV stream file which stores the MPEG-2 transport stream.

When the AV stream has discontinuous points of STC, the same PTS values may appear in the AV stream file. Thus, if a time point in the AV stream is specified on the PTS basis, the PTS pf the access point is insufficient to specify the point. Moreover, an index of the continuous STC domain containing the PTS is required. In this format, the continuous STC domain and its index are termed an STC-sequence and STC-sequence-id, respectively. The STC-sequence information is defined by the STCInfo of the Clip Information file.

The STC-sequence-id is used in an AV stream file and is optional in the AV stream file having the TU_map.

The programs are each a collection of elementary streams and co-own a sole system time base for synchronized reproduction of these streams.

It is useful for a reproducing apparatus (recording and/or reproducing apparatus 1 of Fig.1) to know the contents of an AV stream prior to its decoding. These contents include e.g., values of the PID of a transport packet transmitting an audio or video elementary stream or the type of the video or audio components, such as HDTV video or MPEG-2 AAC audio stream. This information is useful for creating a menu screen foy illustrating to the user the contents of the PlayList referencing the AV stream. It is similarly useful for setting the initial state of the AV decoder and the demultiplexer of the respective apparatus.

For this reason, the Clip Information file owns ProgramInfo for illustrating the program contents.

It may be an occurrence that program contents be changed in the AV stream file in which the MPEG-2 transport stream is stored. For example, the PID of the transport packet transmitting the video elementary stream may be changed, or the component type of the video stream may be changed from SDTV to HDTV.

The ProgramInfo stores the information on change points of program contents in the AV stream file. The domain of the AV stream file in which the program contents remain constant is termed program-sequence.

This program-sequence is used in an AV stream file having EP_map and is optional in an AV stream file having TU_map.

The present embodiment defines the self-encoding stream format (SESF). The SESF is used for encoding analog input signals and for decoding digital input signals for subsequently encoding the decoded signal into an MPEG-2 transport stream.

The SESF defines an elementary stream pertinent to the MPEG-2 transport stream and the AV stream. When the recording and/or reproducing apparatus 1encodes and records input signals in the SESF, an EP_map is created and recorded on the disc.

A digital broadcast stream uses one of the following systems for recording on the recording medium 100: First, the digital broadcast stream is transcoded into an SESF stream. In this case, the recorded stream must conform to SESF and EP_map must be prepared and recorded on the disc.

Alternatively, an elementary stream forming a digital broadcast stream is transcoded to a new elementary stream and re-multiplexed to a new transport stream conforming to the stream format prescribed by the organization for standardizing the digital broadcast stream. In this case, an EP_map must be created and recorded on the disc.

For example, it is assumed that the input stream is an MPEG-2 transport stream conforming to the ISDB (standard appellation of digital BS of Japan), with the transport stream containing the HDTV video stream and the MPEG AAC audio stream. The HDTV video stream is transcoded to an SDTV video stream, which

SDTV video stream and the original AAC audio stream are re-multiplexed to TS. The SDTV stream and the transport stream both need to conform to the ISDB format.

Another system of recording the digital broadcast stream on the recording medium 100 is to make transparent recording of the input transport stream, that is to record the input transport stream unchanged, in which case the EP_map is formulated and recorded on the disc.

Alternatively, the input transport stream is recorded transparently, that is an input transport stream is recorded unchanged, in which case TU_map is created and recorded on the disc.

The directory and the file are hereinafter explained. The recording and/or reproducing apparatus 1 is hereinafter described as DVR (digital video recording). Fig.14 shows a typical directory structure on the disc. The directories of the disc of the DVR may be enumerated by a root directory including "DVR" directory, and the "DVR" directory, including "PLAYLIST" directory, "CLIPINF" directory, "M2TS" directory and "DATA" directory, as shown in Fig.14. Although directories other than these may be created below the root directory, these are discounted in the application format of the present embodiment.

Below the "DATA" directory, there are stored all files and directories prescribed by the DVR application format. The "DVR" directory includes four directories. Below the "PLAYLIST" directory are placed Real PlayList and Virtual PlayList database files. The latter directory may exist in a state devoid of PlayList.

Below "CLIPINF" is placed a Clip database. This directory, too, may exist in a state devoid of AV stream files. In the "DATA" directory, there are stored files of data broadcast, such as digital TV broadcast.

The "DVR" directory stores the following files: That is, an "info.dvr" is created below the DVR directory to store the comprehensive information of an application layer. Below the DVR directory, there must be a sole info.dvr. The filename is assumed to be fixed to info.dvr. The "menu.thmb" stores the information pertinent to the menu thumbnails. Below the DVR directory, there must be 0 or 1 mark thumbnail. The filename is assumed to be fixed to "menu.thmb". If there is no menu thumbnail, this file may not exist.

The "mark.thmb" file stores the information pertinent to the mark thumbnail picture. Below the DVR directory, there must be 0 or 1 mark thumbnail. The filename is assumed to be fixed to "menu.thmb". If there is no menu thumbnail, this file may not exist.

The "PLAYLIST" directory stores two sorts of the PlayList files which are Real PlayList and Virtual PlayList. An "xxxxx.rpls" file stores the information pertinent to one Real PlayList. One file is created for each Real PlayList. The filename is "xxxxx.rpls", where "xxxxx" denotes five numerical figures from 0 to 9. A file extender must be "rpls".

The "yyyyy.vpls" stores the information pertinent to one Virtual PlayList. One file with a filename "yyyyy,vpls" is created from one Virtual PlayList to another,

where "yyyyy" denotes five numerical figures from 0 to 9. A file extender must be "vpls".

The "CLIPINF" directory stores one file in association with each AV stream file. The "zzzzz.clpi" is a Clip Information file corresponding to one AV stream file (Clip AV stream file or Bridge-Clip stream file). The filename is "zzzzz.clpi", where "zzzzz" denotes five numerical figures from 0 to 9. A file extender must be "clpi".

The "M2TS" directory stores an AV stream file. The "zzzzz.m2ts" file is an AV stream file handled by the DVR system. This is a Clip AV stream file or a Bridge-Clip AV stream file. The filename is "zzzzz.m2ts", where "zzzzz" denotes five numerical figures from 0 to 9. A file extender must be "m2ts".

The "DATA" directory stores data transmitted from data broadcasting. This data may, for example, be XML or MPEG files.

The syntax and the semantics of each directory (file) are now explained. Fig.15 shows the syntax of the "info.dvr" file. The "info.dvr" file is made up of three objects, that is DVRVoume(), TableOfPlayLists() and MakersPrivateData().

The syntax of info.dvr shown in Fig.15 is explained. The TableOfPlayLists_Start_address indicates the leading address of the TableOfPlayLists() in terms of the relative number of bytes from the leading byte of the "info.dvr" file. The relative number of bytes is counted beginning from 0.

The MakersPrivateData_Start_address indicates the leading address of the MakersPrivateData(), in terms of the relative number of bytes as from the leading byte

of the "info.dvr" file. The relative number of bytes is counted from 0. The padding_word is inserted in association with the syntax of "info.dvr". N1 and N2 are optional positive integers. Each padding word may assume an optional value.

The DVRVolume() stores the information stating the contents of the volume (disc). Fig.16 shows the syntax of the DVRVolume. The syntax of the DVRVolume(), shown in Fig.16, is now explained. The version_number indicates four character letters indicting the version numbers of the DVRVolume(). The version_number is encoded to "0045" in association with ISO646.

Length is denoted by 32-bit unsigned integers indicating the number of bytes from directly after the length field to the trailing end of DVRVolume().

The ResumeVolume() memorizes the filename of the Real PlayList or the Virtual PlayList reproduced last in the Volume. However, the playback position when the user has interrupted playback of the Real PlayList or the Virtual PlayList is stored in the resume-mark defined in the PlayListMark() (see Figs.42 and 43).

Fig.17 shows the syntax of the ResumeVolume(). The syntax of the ResumeVolume() shown in Fig.17 is explained. The valid_flag indicates that the resume_PlayList_name field is valid or invalid when this 1-bit flag is set to 1 or 0, respectively.

The 10-byte field of resume_PlayList_name indicates the filename of the Real PlayList or the Virtual PlayList to be resumed.

The UIAppInfoVolume in the syntax of the DVRVolume(), shown in Fig.16,

stores parameters of the user interface application concerning the Volume. Fig.18 shows the syntax of the UIAppInfoVolume, the semantics of which are now explained. The 8-bit field of character_set indicates the encoding method for character letters encoded in the Volume_name field. The encoding method corresponds to the values shown in Fig.19.

The 8-bit field of the name_length indicates the byte length of the Volume name indicated in the Volume_name field. The Volume_name field indicates the appellation of the Volume. The number of bytes of the number of the name_length counted from left of the field is the number of valid characters and indicates the volume appellation. The values next following these valid character letters may be any values.

The Volume_protect_flag is a flag indicating whether or not the contents in the Volume can be shown to the user without limitations. If this flag is set to 1, the contents of the Volume are allowed to be presented (reproduced) to the user only in case the user has succeeded in correctly inputting the PIN number (password). If this flag is set to 0, the contents of the Volume are allowed to be presented to the user even in case the PIN number is not input by the user.

If, when the user has inserted a disc into a player, this flag has been set to 0, or the flag is set to 1 but the user has succeeded in correctly inputting the PIN number, the recording and/or reproducing apparatus 1 demonstrates a list of the PlayList in the disc. The limitations on reproduction of the respective PlayLists are irrelevant to the Volume_protect_flag and are indicated by playback_control_flag defined in the

UIAppInfoVolume.

The PIN is made up of four numerical figures of from 0 to 9, each of which is coded in accordance with ISO/IEC 646. The ref_thumbnail_index field indicates the information of a thumbnail picture added to the Volume. If the ref_thumbnail_index field is of a value other than 0xFFFF, a thumbnail picture is added to the Volume. The thumbnail picture is stored in a menu.thumb file. The picture is referenced using the value of the ref_thumbnail_index in the menu.thumb file. If the ref_thumbnail_index field is 0xFFFF, it indicates that a thumbnail picture has been added to the Volume.

The TableOfPlayList() in the info.dvr syntax shown in Fig.15 is explained. The TableOfPlayList() stores the filename of the PlayList (Real PlayList and Virtual PlayList). All PlayList files recorded in the Volume are contained in the TableOfPlayList(), which TableOfPlayList() indicates the playback sequence of the default of the PlayList in the Volume.

Fig.20 shows the syntax of the TableOfPlayList(), which is now explained. The version_number of the TableOfPlayList() indicates four character letters indicating the version numbers of the TableOfPlayLists. The version_number must be encoded to "0045" in accordance with ISO 646.

Length is a unsigned 32-bit integer indicating the number of bytes of the TableOfPlayList() from directly after the length field to the trailing end of the TableOfPlayList(). The 16-bit field of the number_of_PlayLists indicates the number of loops of the for-loop inclusive of the PlayList_file_name. This numerical figure

must be equal to the number of PlayLists recorded in the Volume. The 10-byte numerical figure of the PlayList_file_name indicates the filename of the PlayLists.

Fig.21 shows another configuration of the syntax of the TableOfPlayList(). The syntax shown in Fig.21 is comprised of the syntax shown in Fig.20 in which is contained the UIAppInfoPlayList. By such structure including the UIAppInfoPlayList, it becomes possible to create a menu picture simply on reading out the TableOfPlayLists. The following explanation is premised on the use of the syntax shown in Fig.20.

The MakersPrivateData in the info.dvr shown in Fig.15 is explained. The MakersPrivateData is provided to permit the maker of the recording and/or reproducing apparatus 1 to insert private data of the maker in the MakersPrivateData() for special applications of different companies. The private data of each maker has standardized maker_ID for identifying the maker who has defined it. The MakersPrivateData() may contain one or more maker_ID.

If a preset maker intends to insert private data, and the private data of a different maker is already contained in the MakersPrivateData(), the new private data is added to the MakersPrivateData() without erasing the pre-existing old private data. Thus, in the present embodiment, private data of plural makers can be contained in one MakersPrivateData().

Fig.22 shows the syntax of the MakersPrivateData. The syntax of the MakersPrivateData shown in Fig.22 is explained. The version_number of the

TableOfPlayList() indicates four character letters indicating the version numbers of the TableOfPlayLists. The version_number must be encoded to "0045" in accordance with ISO 646. Length is a unsigned 32-bit integer indicating the number of bytes of the TableOfPlayList() from directly after the length field to the trailing end of the MakersPrivateData().

The mpd_blocks_start_address indicates the leading end address of the first mpd_block() in terms of the relative number of bytes from the leading byte of the MakersPrivateData(). The number_of_maker_entries is the 16-bit codeless integer affording the number of entries of the maker private data included in the MakersPrivateData(). There must not be present two or more maker private data having the same maker_ID values in the MakersPrivateData().

The mpd_blocks_size is a 16-bit unsigned integer affording one mpd_block size in terms of 1024 bytes as a unit. For example, if mpd_block_size = 1, it indicates that the size of one mpd_block is 1024 bytes. The number_of_mpd_blocks is a 16-bit unsigned integer affording the number of mpd_blocks contained in the MakersPrivateData(). The maker_ID is the 16-bit unsigned integer indicating the model number code of the DVR system which has created the maker private data. The value encoded to the maker_ID is specified by the licensor.

The maker_model_code is a 16-bit unsigned integer indicating the model number code of the DVR system which has created the maker private data. The value encoded to the maker_model_code is set by the maker who has received the license

of the format. The start_mpd_block_number is a 16-bit unsigned integer indicating the number of the mpd_block number at which begins the maker private data. The leading end of the maker private data must be aligned with the leading end of the mpd_block. The start_mpd_block_number corresponds to a variable j in the for-loop of the mpd_block.

The mpd_length is a 32-bit unsigned integer indicating the size of the maker private data. The mpd_block is an area in which is stored maker private data. All of the mpd_blocks in the MakersPrivateData() must be of the same size.

The real PlayList file and the Virtual PlayList file, in other words, xxxxx.rpls and yyyyy.vpls, are explained. Fig.23 shows the syntax of xxxxx.rpls (Real PlayList) and yyyyy.vpls (Virtual PlayList), which are of the same syntax structure. Each of the xxxxx.rpls and yyyyy.vpls is made up of three objects, that is PlayList(), PlayListMark() and MakersPrivateData().

The PlayListMark_Start_address indicates the leading address of the PlayListMark(), in terms of the relative number of bytes from the leading end of the PlayList file as a unit. The relative number of bytes is counted from zero.

The MakersPrivateData_Start_address indicates the leading address of the MakersPrivateData(), in terms of the relative number of bytes from the leading end of the PlayList file as a unit. The relative number of bytes is counted from zero.

The padding_word (padding word) is inserted in accordance with the syntax of the PlayList file, with N1 and N2 being optional positive integers. Each padding word

may assume an optional value.

PlayList will be further explained in the following although it has been explained briefly. A playback domain in all Clips except Bridge-Clip must be referred by all PlayLists in the disc. Also, two or more Real PlayLists must not overlap the playback domains shown by their PlayItems in the same Clip.

Reference is made to Figs.24A, 24B and 24C. For all Clips, there exist corresponding Real PlayLists, as shown in Fig.24A. This rule is observed even after the editing operation has come to a close, as shown in Fig.24B. Therefore, all Clips must be viewed by referencing one of Real PlayLists.

Referring to Fig.24C, the playback domain of the Virtual PlayList must be contained in the playback domain and in the Bridge-Clip playback domain. There must not be present in the disc Bridge-Clip not referenced by any Virtual PlayList.

The Real PlayList, containing the list of the PlayItem, must not contain SubPlayItem. The Virtual PlayList contains the PlayItem list and, if the CPI_type contained in the PlayList() is the EP_map type and the PlayList_type is 0 (PlayList containing video and audio) , the Virtual PlayList may contain one SubPlayItem. In the PlayList() in the present embodiment, the SubPlayItem is used only for audio post recording. The number of the SubPlayItems owned by one Virtual PlayList must be 0 or 1.

The PlayList is hereinafter explained. Fig.25 shows the PlayList syntax which is now explained. The version_number indicates four character letters indicting the

version numbers of the PlayList(). The version_number is encoded to "0045" in association with ISO 646. Length is a 32-bit unsigned integer indicating the total number of byte of the PlayList() as from directly after the length field to the trailing end of the PlayList(). The PlayList_type, one example of which is shown in Fig.26, is an 8-bit field indicating the PlayList type.

The CPI_type is one-bit flag indicating the value of the CPI_type of the Clip referenced by the PlayItem() and by the SubPlayItem(). The CPI_types defined in the CPIs of all Clips referenced by one PlayList must be of the same values. The number_of_PlayItems is a 16-bit field indicating the number of PlayItems present in the PlayList.

The PlayItem_id corresponding to the preset PlayItem() is defined by the sequence in which the PlayItem() appears in the for-loop containing the PlayItem(). The PlayItem_id begins with 0. The nimber_of_SubPlayItems is a 16-bit field indicating the number of SubPlayItem in the PlayList. This value is 0 or 1. An additional audio stream path (audio stream path) is a sort of a sub path.

The UIAppInfoPlayList of the PlayList syntax shown in Fig.25 is explained. The UIAppInfoPlayList stores parameters of the user interface application concerning the PlayList. Fig.27 shows the syntax of the UIAppInfoPlayList, which is now explained. The character_set is an 8-bit field indicating the method for encoding character letters encoded in the PlayList_name field. The encoding method corresponds to the values conforming to the table shown in Fig.19.

The name_length is an 8-bit field indicating the byte length of the PlayList name indicated in the PlayList_name field. The PlayList_name field shows the appellation of the PlayList. The number of bytes of the number of the name_length counted from left of the field is the number of valid characters and indicates the PlayList appellation. The values next following these valid character letters may be any values.

The record_time_and_date is a 56-bit field storing the date and time on which the PlayList was recorded. This field is 14 numerical figures for year/ month/ day/ hour/minute/ second encoded in binary coded decimal (BCD). For example, 2001/ 12/ 23:01:02:03 is encoded to "0x20011223010203".

The duration is a 24-bit field indicating the total replay time of the PlayList in terms of hour/ minute/ second as a unit. This field is six numerical figures encoded in binary coded decimal (BCD). For example, 01:45:30 is encoded to "0x014530".

The valid_period is a 32-bit field indicating the valid time periods of the PlayList. This field is 8 numerical figures encoded in 4-bit binary coded decimal (BCD). The valid_period is used in the recording and/or reproducing apparatus 1 e.g., when the PlayList, for which the valid period has lapsed, is to be automatically erased. For example, 2001/05/07 is encoded to "0x20010507".

The maker_ID is a 16-bit unsigned integer indicating the maker of the DVR player (recording and/or reproducing apparatus 1) which has been the last to update its PlayList. The value encoded to maker_ID is assigned to the licensor of the DVR format. The maker_code is a 16-bit unsigned integer indicating the model number of

the DVR player which has been the last to update the PlayList. The value encoded to the maker_code is determined by the maker who has received the license of the DVR format.

If the flag of the playback_control_flag is set to 1, its PlayList is reproduced only when the user successfully entered the PIN number. If this flag is set to 0, the user may view the PlayList without the necessity of inputting the PIN number.

If the write_protect_flag is set to 1, the contents of the PlayList are not erased nor changed except the write_protect_flag. If this flag is set to 0, the user is free to erase or change the PlayList. If this flag is set to 1, the recording and/or reproducing apparatus 1 demonstrates a message requesting re-confirmation by the user before the user proceeds to erase, edit or overwrite the PlayList.

The Real PlayList, in which the write_protect_flag is set to 0, may exist, the Virtual PlayList, referencing the Clip of the Real PlayList may exist, and the write_protect_flag of the Virtual PlayList may be set to 1. If the user is desirous to erase the Real PlayList, the recording and/or reproducing apparatus 1 issues an alarm to the user as to the presence of the aforementioned Virtual PlayList or "minimizes" the Real PlayList before erasing the Real PlayList.

If is_played_flag is set to 1, as shown in Fig.28B, it indicates that the PlayList was reproduced at least once since it was recorded, whereas, if it is set to 0, it indicates that the PlayList was not reproduced even once since it was recorded.

Archive is a two-bit field indicating whether the PlayList is an original or a

copy, as shown in Fig.28C. The field of ref_thumbnail_index indicates the information of a thumbnail picture representative of the PlayList. If the ref_thumbnail_index field is of a value other than 0xFFFF, a thumbnail picture representative of the PlayList is added in the PlayList, with the PlayList being stored in the menu.thmb file. The picture is referenced using the value of ref_thumbnail_index in the menu.thmb file. If the ref_thumbnail_index field is 0xFFFF, no thumbnail picture representative of the PlayList is added in the PlayList.

The PlayItem is hereinafter explained. One PlayItem() basically contains the following data: Clip_Information_file_name for specifying the filename of the Clip, IN-time and OUT-time, paired together to specify the playback domain of Clip, STC_sequence_id referenced by IN-time and OUT-time in case the CPI_type defined in PlayList() is EP_map type, and Connection_Condition indicating the connection condition of previous PlayItem and current PlayItem.

If PlayList is made up of two or more PlayItems, these PlayItems are arrayed in a row, without temporal gap or overlap, on the global time axis of the PlayList. If CPI_type defined in the PlayList is EP_map type and the current PlayList does not have the BridgeSequence(), the IN-time and OUT-time pair must indicate the same time on the STC continuous domain as that specified by the STC_sequence_id. Such instance is shown in Fig.29.

Fig.30 shows such a case in which the CPI_type defined by PlayList() and, if the current PlayItem has the BridgeSequence(), the rules as now explained are applied.

The IN_time of the PlayItem previous to the current PlayItem, shown as IN_time1, indicates the time in Bridge-Clip specified in the BridgeSequenceInfo() of the current PlayItem. This OUT_time must obey the encoding limitations which will be explained subsequently.

The IN_time of the current PlayItem, shown as IN_time2, indicates the time in Bridge-Clip specified in the BridgeSequenceInfo() of the current PlayItem. This IN_time also must obey the encoding limitations as later explained. The OUT_time of PlayItem of the current PlayItem, shown as OUT_time2, indicates the time on the STC continuous domain specified by STC_sequenbce_id of the current PlayItem.

If the CPI_type of PlayList() is TU_map type, the IN_time and OUT_time of PlayItem, paired together, indicate the time on the same Clio AV stream, as shown in Fig.31.

The PlayItem syntax is as shown in Fig.32. As to the syntax of the PlayItem, shown in Fig.32, the field of the Clip_information_file_name indicates the filename of the Clip Information. The Clip_stream_type defined by the ClipInfo() of this Clip Information file must indicate the Clip AV stream.

The STC_sequence_id is an 8-bit field and indicates the STC_sequence_id of the continuous STC domain referenced by the PlayItem. If the CPI_type specified in the PlayList() is TU_map type, this 8-bit field has no meaning and is set to 0. IN_time is a 32-bit field and used to store the playback start time of PlayItem. The semantics of IN_time differs with CPI_type defined in the PlayList(), as shown in Fig.33.

OUT_time is a 32-bit field and is used to store the playback end time of PlayItem. The semantics of OUT_time differs with CPI_type defined in the PlayList(), as shown in Fig.34.

Connection_condition is a 2-bit field indicating the connection condition between the previous PlayItem and the current PlayItem, as shown in Fig.35. Figs.36A to 36D illustrate various states of Connection_condition shown in Fig.35.

BridgeSequenceInfo is explained with reference to Fig.37. This BridgeSequenceInfo is the ancillary information of the current PlayItem and includes the following information. That is, BridgeSequenceInfo includes Bridge_Clip_Information_file_name for specifying the Bridge_Clip AV file and a Bridge_Clip_Information_file_name specifying the corresponding Clip Information file (Fig.45).

It is also an address of a source packet on the Clip AV stream referenced by the previous PlayItem. Next to this source packet is connected the first source packet of the Bridge-Clip AV stream. This address is termed the RSPN_exit_from_previous_Clip. It is also an address of the source packet on the Clip AV stream referenced by the current PlayItem. Ahead of this source packet is connected the last source packet of the Bridge_clip AV stream file. This address is termed RSPN_enter_to_current_Clip.

In Fig.37, RSPN_arrival_time_discontinuity indicates an address of a source packet in the Bridge_Clip AV stream where there is a discontinuous point in the

arrival time base. This address is defined in the ClipInfo() (Fig.46).

Fig.38 shows the syntax of the BridgeSequenceInfo. Turning to the syntax of BridgeSequenceInfo shown in Fig.38, the field of Bridge_Clip_Information_file_name indicates the filename of the Clip Information file corresponding to the Bridge_Clip_Information_file. The Clip_stream_type defined in ClipInfo() of this Clip information file must indicate 'Bridge_Clip AV stream'.

The 32-bit field of the RSPN_exit_from_previous_Clip is a relative address of a source packet on the Clip AV stream referenced by the previous PlayItem. Next to this source packet is connected the first source packet of the Bridge_Clip AV stream file. The RSPN_exit_from_previous_Clip has a size based on the source packet number as a unit, and is counted with the value of the offset_SPN defined in the ClipInfo() from the first source packet of the Clip AV stream file referenced by the previous PlayItem.

The 32-bit field of RSPN_enter_to_curent_Clip is the relative address of the source packet on the Clip AV stream referenced by the current PlayItem. Ahead of this source packet is connected the last source packet of the Bridge_Clip_AV stream file. The RSPN_enter_to_curent_Clip has a size that is based on the source packet number as a unit. The RSPN_enter_to_curent_Clip is counted with the value of the offset_SPN, defined in the ClipInfo() from the first source packet of the Clip AV stream file referenced by the current PlayItem, as an initial value.

The SubPlayItem is explained with reference to Fig.39. The use of

SubPlayItem() is permitted only if the CPI_type of the PlayList() is the EP_map type. In the present embodiment, SubPlayItem is used only for audio post recording. The SubPlayItem() includes the following data. First, it includes Clip_Information_file_name for specifying the Clip referenced by the sub path in the PlayList.

It also includes SubPath_IN_time and SubPath_OUT_time for specifying the sub path playback domain in the Clip. Additionally, it includes sync_PlayItem_id and start_PTS_of_PlayItem for specifying the time of starting the sub path reproduction on the main path time axis. The Clip AV stream, referenced by the sub path, must not contain STC discontinuous points (discontinuous points of the system time base). The clocks of audio samples of the Clip used in the sub path are locked to the clocks of the audio samples of the main path.

Fig.40 shows the syntax of the SubPlayItem. Turning to the syntax of the SubPlayItem, shown in Fig.40, the field of the Clip_Information_file_name indicates the filename of the Clip Information file and is used by a sub path in the PlayList. The Clip_stream_type defined in this ClipInfo() must indicate the Clip AV stream.

An 8-bit field of sync_PlayItem_id indicates the sub path type. Here, only '0x00' is set, as shown in Fig.41, while other values are reserved for future use.

The 8-bit field of sync_PlayItem_id indicates the PlayItem_id of the PlayItem containing the time of playback start of the sub path on the time axis of the main path. The value of PlayItem_id corresponding to the preset PlayItem is defined in the

PlayList() (Fig.25).

A 32-bit field of sync_start_PTS_of_PlayItem denotes the time of playback start of the sub path on the time axis of the main path, and denotes the upper 32 bits of the PTS (presentation time stamp) on the PlayItem referenced by the sync_PlayItem_id. The upper 32 bit field of the SubPath_IN_time stores the playback start time of the sub path. SubPath_IN_time denotes upper 32 bits of the PTS of 33 bits corresponding to the first presentation unit in the sub path.

The upper 32 bit field of subPath_OUT_time stores the playback end time of the sub path. SubPath_OUT_time indicates upper 32 bits of the value of the Presentation_end_TS calculated by the following equation:

Presentation_end_TS = PTS_OUT + AU_duration

where PTS_out is the PTS of the 33 bit length corresponding to the last presentation unit of the SubPath and AU_duration is the 90 kHz based display period of the last presentation unit of the SubPath.

Next, PlayListMark() in the syntax of xxxxx.rpls and yyyyy.vpls shown in Fig.23 is explained. The mark information pertinent to the PlayList is stored in this PlayListMark. Fig.42 shows the syntax of PlayListMark. Turning to the syntax of the PlayListMark shown in Fig.42, version_number is four character letters indicating the version number of this PlayListMark(). The version_number must be encoded to "0045" in accordance with ISO 646.

Length is an unsigned 32-bit integer indicating the number of bytes of

PlayListMark() from directly after the length field to the trailing end of the PlayListMark(). The number_of_PlayListMarks is a 16-bit unsigned integer indicating the number of marks stored in the PlayListMark. The number_of_PlayListMarks may be zero. The mark_type is an 8-bit field indicating the mark type and is encoded in the table shown in Fig.43.

A 32-bit filed of mark_time_stamp stores a time stamp indicating the point specified by the mark. The semantics of the mark_time_stamp differ with CPI_type defined in the PlayList(), as shown in Fig.44. The PlayItem_id is an 8-bit field specifying the PlayItem where the mark is put. The values of PlayItem_id corresponding to a preset PlayItem is defined in the PlayList() (see Fig.25).

An 8-bit field of character_set shows the encoding method of character letters encoded in the mark_name field. The encoding method corresponds to values shown in Fig.19. The 8-bit field of name_length indicates the byte length of the mark name shown in the mark_name field. The mark_name field denotes the mark name indicated in the mark_name field. The number of bytes corresponding to the number of name_lengths from left of this field is the effective character letters and denotes the mark name. In the mark_name field, the value next following these effective character letters may be arbitrary.

The field of the ref_thumbnail_index denotes the information of the thumbnail picture added to the mark. If the field of the ref_thumbnail_index is not 0xFFFF, a thumbnail picture is added to its mark, with the thumbnail picture being stored in the

mark.thmb file. This picture is referenced in the mark.thmb file, using the value of ref_thumbnail_index, as explained subsequently. If the ref_thumbnail_index field is 0xFFFF, it indicates that no thumbnail picture is added to the mark.

The Clip Information file is now explained. The zzzzz.clpi (Clip Information file) is made up of six objects, as shown in Fig.45. These are ClipInfo(), STC_Info(), Program(), CPI(), ClipMark() and MakersPrivateData(). For the AV stream (Clip AV stream or Bridge-Clip AV stream) and the corresponding Clip Information file, the same string of numerals "zzzzz" is used.

Turning to the syntax of zzzzz.clpi (Clip Information file) shown in Fig.45 is explained. The ClipInfo_Start_address indicates the leading end address of ClipInfo() with the relative number of bytes from the leading end byte of the zzzzz.clpi file as a unit. The relative number of bytes is counted from zero.

The STC_Info_Start_address indicates the leading end address of STC_Info with the relative number of bytes from the leading end byte of the zzzzz.clpi file as a unit. The ProgramInfo_Start_address indicates the leading end address of ProgramInfo() with the relative number of bytes from the leading end byte of the zzzzz.clpi file as a unit. The relative number of bytes is counted from 0. The CPI_Start_address indicates the leading end address of CPI() with the relative number of bytes from the leading end byte of the zzzzz.clpi file as a unit. The relative number of bytes is counted from zero.

The ClipMark_Start_address indicates the leading end address of ClipMark()

with the relative number of bytes from the leading end byte of the zzzzz.clpi file as a unit. The relative number of bytes is counted from zero. The_MakersPrivateData Start_address indicates the leading end address of MakersPrivateData() with the relative number of bytes from the leading end byte of the zzzzz.clpi file as a unit. The relative number of bytes is counted from zero. The padding_word is inserted in accordance with the syntax of the zzzzz.clpi file. N1, N2, N3, N4 and N5 must be zero or optional positive integers. The respective padding words may also assume optional values.

The ClipInfo is now explained. Fig.46 shows the syntax of ClipInfo. Fig.46 shows the syntax of ClipInfo. In the ClipInfo() is stored the attribute information of corresponding AV stream files (Clip AV stream or Bridge-Clip AV stream file).

Turning to the syntax of the ClipInfo shown in Fig.46, version_number is the four character letters indicating the version number of this ClipInfo(). The version_number must be encoded to "0045" in accordance with the ISO 646. Length is a 32-bit unsigned integer indicating the number of bytes of ClipInfo() from directly at back of the length field to the trailing end of the ClipInfo(). An 8-bit field of Clip_stream_type indicates the type of the AV stream corresponding to the Clip Information file, as shown in Fig.47. The stream types of the respective AV streams will be explained subsequently.

The 32-bit field of offset_SPN gives an offset value of the source packet number of the first source packet number of the first source packet of the AV stream

(Clip AV stream or the Bridge-Clip AV stream). When the AV stream file is first recorded on the disc, this offset_SPN must be zero.

Referring to Fig.48, when the beginning portion of the AV stream file is erased by editing, offset_SPN may assume a value other than 0. In the present embodiment, the relative source packet number (relative address) referencing the offset_SPN is frequently described in the form of RSPNxxx, where xxx is modified such that RSPN_xxx is RAPN_EP_start. The relative source packet number is sized with the source packet number as a unit and is counted from the first source packet number of the AV stream file with the value of the offset_SPN as the initial value.

The number of source packets from the first source packet of the AV stream file to the source packet referenced by the relative source packet number (SPN_xxx) is calculated by the following equation:

SPN_xxx = RSPN_xxx – offset_SPN.

Fig.48 shows an instance in which offset_SPN is 4.

TS_recording_rate is a 24-bit unsigned integer, which affords an input/output bit rate required for the AV stream to the DVR drive (write unit 22) or from the DVR drive (readout unit 28). The record_time_and_date is a 56-bit field for storing the date and time of recording of the AV stream corresponding to the Clip and is an encoded representation of year/month/day/hour/minute in 4-bit binary coded decimal (BCD) for 14 numerical figures. For example, 2001/2/23:01:02:03 is encoded to "0x20011223010203".

The duration is a 24-bit field indicating the total playback time of the Clip by hour/minute/second based on arrival time clocks. This field is six numerical figures encoded in 4-bit binary coded decimal (BCD). For example, 01:45:30 is encoded to "0x014530".

A flag time_controlled_flag indicates the recording mode of an AV stream file. If this time_controlled_flag is 1, it indicates that the recording mode is such a mode in which the file size is proportionate to the time elapsed since recording, such that the condition shown by the following equation:

$$Ts\_average\_rate*192/188*(t - start\_time) - \alpha <= size\_clip(t)$$
$$<= TS\_average\_rate*192/188*(t - start\_time) + \alpha$$

where TS_average_rate is an average bit rate of the transport stream of the AV stream file expressed by bytes/second.

In the above equation, t denotes the time in seconds, while start_time is the time point when the first source packet of the AV stream file was recorded. The size_clip(t) is 10*192 bytes and α is a constant dependent on TS_average_rate.

If time_controlled_flag is set to 0, it indicates that the recording mode is not controlling so that the time lapse of recording is proportionate to the file size of the AV stream. For example, the input transport stream is recorded in a transparent fashion.

If time_controlled_flag is set to 1, the 24-bit field of TS_average_rate indicates the value of TS_average_rate used in the above equation. If time_controlled_flag is

set to 0, this field has no meaning and must be set to 0. For example, the variable bit rate transport stream is encoded by the following sequence: First, the transport rate is set to the value of TS_recording_rate. The video stream is encoded with a variable bit rate. The transport packet is intermittently encoded by not employing null packets.

The 32-bit field of RSPN_arrival_time_discontinuity is a relative address of a site where arrival timebase discontinuities are produced on the Bridge-Clip AV stream file. The RSPN_arrival_time_discontinuity is sized with the source packet number as a unit and is counted with the value of offset_SPN defined in the ClipInfo() as from the first source packet of the Bridge-Clip AV stream file. An absolute address in the Bridge-Clip AV stream file is calculated based on the aforementioned equation:

SPN_xxx = RSPN_xxx – offset_SPN.

The 144-bit field of reserver_for_system_use is reserved for a system. If is_format_identifier_valid flag is 1, it indicates that the field of format_identifier is effective. If is_format_identifier_valid flag is 1, it indicates that the format_identifier field is valid. If is_original_network_ID_valid flag is 1, it indicates that the field of is_transport_stream_ID-valid is valid. If the flag is_transport_stream_ID-valid is 1, it indicates that the transport_stream_ID field is valid. If is_servece_ID_valid flag is 1, it indicates that the servece_ID field is valid.

If is_country_code_valid flag is 1, it indicates that the field country_code is valid. The 32-bit field of format_identifier indicates the value of format_identifier owned by a registration descriptor (defined in ISO/IEC13818-1) in the transport

stream. The 16-bit field of original_network_ID indicates the value of the original_network_ID defined in the transport stream.

The 16-bit field in servece_ID denotes the value of servece_ID defined in the transport stream. The 24-bit field of country_code shows a country code defined by ISO3166. Each character code is encoded by ISO8859-1. For example, Japan is represented as "JPN" and is encoded to "0x4A 0x50 0x4E". The stream_format_name is 15 character codes of ISO-646 showing the name of a format organization affording stream definitions of transport streams. An invalid byte in this field has a value of '0xFF'.

Format_identifier, original_network_ID, transport_stream_ID, servece_ID, country_code and stream_format_name indicate service providers of transport streams. This allows to recognize encoding limitations on audio or video streams and stream definitions of private data streams other than audio video streams or SI (service information). These information can be used to check if the decoder is able to decode the stream. If such decoding is possible, the information may be used to initialize the decoder system before starting the decoding.

STC_Info is now explained. The time domain in the MPEG-2 transport stream not containing STC discontinuous points(discontinuous points of the system time base) is termed the STC_sequence. In the Clip, STC_sequence is specified by the value of STC_sequence_id. Figs.50A and 50B illustrate a continuous STC domain. The same STC values never appear in the same STC_sequence, although the maximum tine

length of Clip is limited, as explained subsequently. Therefore, the same PTS values also never appear in the same STC_sequence. If the AV stream contains N STC discontinuous points, where N > 0, the Clip system time base is split into (N+1) STC_sequences.

STC_Info stores the address of the site where STC discontinuities (system timebase discontinuities) are produced. As explained with reference to Fig.51, the RSPN_STC_start indicates the address and begins at a time point of arrival of the source packet referenced by the (k+1)st RSPN_STC_start and ends at a time point of arrival of the last source packet.

Fig.52 shows the syntax of the STC_Info. Turning to the syntax of STC_Info shown in Fig.52, version_number is four character letters indicating the version number of STC_Info(). The version_number must be encoded to "0045" in accordance with ISO 646.

Length is a 32-bit unsigned integer indicating the number of bytes of STC_Info() from directly after this length field to the end of STC_Info. If CPI_type of CPI() indicates TU_map type, 0 may be set in this length field. If CPI_type of CPI() indicates EP_map type, the num_of_STC_sequence mut be of a value not less than 1.

An 8-bit unsigned integer of num_of_STC_sequence indicates the number of sequences in the Clip. This value indicates the number of the for-loops next following the field. The STC_sequence_id corresponding to the preset STC_sequence is defined

by the order in which appears the RSPN_STC_start corresponding to the STC_sequence in the for-loop containing the RSPN_STC_start. The STC_sequence_id commences at 0.

The 32-bit field of RSPN_STC_start indicates an address at which the STC_sequence commences on the AV stream file. RSPN_STC_start denotes an address where system time base discontinuities are produced in the AV stream file. The RSPN_STC_start may also be a relative address of the source packet having the first PCR of the new system time base in the AV stream. The RSPN_STC_start is of a size based on the source packet number and is counted from the first source packet of the AV stream file with the offset_SPN value defined in ClipInfo() as an initial value. In this AV stream file, the absolute address is calculated by the above-mentioned equation, that is

SPN_xxx = RSPN_xxx – offset_SPN.

ProgramInfo in the syntax of zzzz.clip shown in Fig.45 is now explained with reference to Fig.53. The time domain having the following features in the Clip is termed program_sequence. These feature are that the value of PCR_PID is not changed, the number of audio elementary streams is also not changed, the PID values in the respective video streams are not changed, the encoding information which is defined by VideoCodingInfo thereof is not changed, the number of audio elementary streams is also not changed, the PID values of the respective audio streams are not changed, and that the encoding information, which is defined by AudioCodingInfo

thereof, is not changed.

Program_sequence has only one system time base at the same time point. Program_sequence has a sole PMT at the same time point. ProgramInfo() stores the address of the site where the program_sequence commences. RSPN_program_sequence-start indicates the address.

Fig.54 illustrates the syntax of ProgramInfo. Turning to the ProgramInfo shown in Fig.54, version_number is four character letters indicating the version number of ProgramInfo(). The version_number must be encoded to "0045" in accordance with ISO 646.

Length is a 32-bit unsigned integer indicating the number of bytes of ProgramInfo() from directly at back of this length field to the end of program(info(). If CPI_type of CPI() indicates the TU_map type, this length field may be set to 0. If the CPI_type of CPI() indicates EP_map type, the number_of_programs must be of a value not less than 1.

An 8-bit unsigned integer of number_of_program_sequences denotes the number of program_sequences in the Clip. This value indicates the number of for-loops next following this field. If program_sequence in the Clip is not changed, 1 must be set in the number of program_sequences. A 32-bit field of RSPN_program_sequence_start is a relative address where the program sequence commences on the AV stream.

RSPN_program_sequence_start is sized with the source packet number as a unit

and is counted with the value of offset_SPN defined in the ClipInfo() as from the first source packet of the AV stream file. An absolute address in the AV stream file is calculated by:

SPN_xxx = RSPN_xxx – offset_SPN.

The values of RSPN_program_sequence_start in the for-loop syntax must appear in the rising order.

A 16-bit field of PCR_PID denotes the PID of the transport packet containing an effective PCR field effective for the program_sequence. An 8-bit field of number_of_audios indicates the number of for-loops containing audio_stream_PID and AudioCodingInfo(). A 16-bit field of video_stream_PID indicates the PID of the transport packet containing a video stream effective for its program_sequence. VideoCodingInfo(), next following this field, must explain the contents of the video stream referenced by its video_stream_PID.

A 16-bit field of audio_stream_PID indicates the PID of a transport packet containing the audio stream effective for its program_sequence. The AudioCodingInfo(), next following this field, must explain the contents of the video stream referenced by its audio_stream_PID.

The order in which the values of video_stream_PID in the for-loop of the syntax must be equal to the sequence of PID encoding of the video stream in the PMT effective for the program_sequence. Additionally, the order in which the values of audio_stream_PID appears in the for-loop of the syntax must be equal to the sequence

of PID encoding of the audio stream in the PMT effective for the program_sequence.

Fig.55 shows the syntax of VideoCodingInfo in the syntax of the ProgramInfo shown in Fig.54. Turning to the syntax of the VideoCoding Info shown in Fig.55, an 8-bit field of video_format indicates the video format corresponding to video_stream_PID in ProgramInfo(), as shown in Fig.56.

Referring to Fig.57, an 8-bit field of frame_rate indicates the video frame rate corresponding to the video_stream_PID in ProgramInfo(). An 8-bit field of display_aspect_ratio indicates a video display aspect ratio corresponding to video_stream_PID in ProgramInfo().

Fig.59 shows the syntax of AudioCodingInfo in the syntax of ProgramInfo shown in Fig.54. Turning to the syntax of the AudioCoding Info shown in Fig.59, an 8-bit field of audio_format indicates the audio encoding method corresponding to audio_stream_PID in ProgramInfo(), as shown in Fig.60.

An 8-bit field of audio_component_type indicates an audio component type corresponding to audio_stream_PID in ProgramInfo() as shown in Fig.61, whilst an 8-bit field of sampling_frequency indicates an audio sampling frequency corresponding to audio_stream_PID in ProgramInfo() as shown in Fig.62.

The CPI (Characteristics Point Information) in the syntax of zzzzz.clip shown in Fig.45 is explained. The CPI is used for correlating the time information in the AV stream with the address in its file. The CPI is of two types, namely EP_map and TU_map. In Fig.63, if CPI_type in CPI() is EP_map, its CPI() contains EP_map. In

Fig.64, if CPI_type in CPI() is TU_map, its CPI() contains TU_map. One AV stream has one EP_map or one TU_map. If the AV stream is an SESF transport stream, the corresponding Clip must own an EP_map.

Fig.65 show the syntax of CPI. Turning to the syntax of CPI shown in Fig.65, the version_number is four character letters indicating the version number of this CPI(). The version_number must be encoded to "0045" in accordance with ISO 646. Length is a 32-bit unsigned integer indicating the number of bytes as from directly after this length field to the trailing end of the CPI(). The CPI_type is a 1-bit flag and indicates the CPI type of Clip, as shown in Fig.66.

The EP_map in the CPI syntax shown in Fig.65 is explained. There are two types of the EP_map, that is EP_map for a video stream and an EP_map for an audio stream. The EP_map_type in the EP_map differentiates between these EP_map types. If the Clip contains one or more video streams, the EP_map for the video stream must be used. If the Clip does not contain a video stream but contains one or more audio streams, the EP_map for the audio stream must be used.

The EP_map for a video stream is explained with reference to Fig.67. The EP_map for the video stream has data stream_PID, PTS_EP_start and RSPN_EP_start. The stream_PID shows the PID of the transport packet transmitting a video stream. The PTS_EP_start indicates the PTS of an access unit beginning from the sequence header of the video stream. The RSPN_RP_start indicates the address of a source packet including the first byte of the access unit referenced by the

PTS_EP_start in the AV stream.

A sub table, termed EP_map_for_one_stream_PID() is created from one video stream transmitted by the transport packet having the same PID to another. If plural video streams exist in the Clip, the EP_map may contain plural EP_map_for_one_stream_PID().

The EP_map for audio stream has data stream_PID, PTS_EP_start and RSPN_EP_start. The stream_PID shows a PID of a transport packet transmitting an audio stream. The PTS_EP_start shows the PTS of an accessing unit in the audio stream. The RSPN_EP-start indicates an address of a source packet containing a first byte of the access unit referenced by PTS_EP_start of the AV stream.

The sub table termed EP_map_for_one_stream_PID() is created from one audio stream transmitted by the transport packet having the same PID to another. If there exist plural audio streams in the Clip, EP_map may contain plural EP_map-_for_one_stream_PID().

Turning to the relation between EP_map and STC_Info, one EP_map_for_one_stream_PID() is created in one table irrespective of discontinuous points in the STC. Comparison of the value of the RSPN_EP_start to the value of RSPN_STC_start defined in STC_Info() reveals the boundary of data of EP_map belonging to respective STC_sequences (see Fig.68). The EP_map must have one EP_map_for_one_stream_PID for a continuous stream range transmitted by the same PID. In the case shown in Fig.69, program#1 and program#3 have the same video

PID, however, the data range is not continuous, so that EP_map_for_one_stream_PID must be provided for each program.

Fig.70 shows the EP_map syntax. By way of explanation of the EP_map syntax shown in Fig.70, the EP_type is a 4-bit field and shows the EP_map entry point type, as shown in Fig.71. The EP_type shows the semantics of the data field next following this field. If Clip includes one or more video stream, the EP_type must be set to 0 ('video'). Alternatively, if the Clip contains no video stream but contains one or more audio stream, then EP_type must be set to 1 ('audio').

The 16-bit field of number_of_stream_PIDs indicates the number of times of loops of the for-loop having number_of_stream_PIDs in the EP_map() as a variable. The 16-bit field of stream_PID(k) indicates the PID of the transport packet transmitting the number k elementary stream (video or audio stream) referenced by EP_map_for_one_stream_PID (num_EP_entries(k)). If EP_type is 0 ('video'), its elementary stream must be a video stream. If EP_type is equal to 1 ('audio'), its elementary stream must be the audio stream.

The 16-bit field of num_EP_entries(k) indicates the num_EP_entries(k) referenced by EP_map_entries(k)). The EP_map_for_one_stream_PID_ Start_address(k): This 32-bit field indicates the relative address position at which the EP_map_for_one_stream_ PID(num_EP_entries(k)) begins in the EP_map(). This value is indicated by the size as from the first byte of the EP_map().

Padding_word must be inserted in accordance with the EP_map() syntax. X and

Y must be optional positive integers. The respective padding words may assume any optional values.

Fig.72 shows the syntax of EP_map_for_one_stream_PID. By way of explanation of the syntax of the EP_map_for_one_stream_PID shown in Fig.72, the semantics of the 32-bit field of PTS_EP_start differs with the EP_type defined by EP_map(). If EP_type is equal to 0 ('video'), this field has upper 32 bits of the 33-bit precision PTS of the access unit beginning with a sequence header of the video stream. If the EP_type is equal to 1 ('audio'), this field has upper 32 bits of PTS of 33 bit precision of the access unit of the audio stream.

The semantics of the 32-bit field of RSPN_EP_start differs with the EP_type defined in EP_map(). If EP_type is equal to 0 ('video'), this field indicates the relative address of the source packet including the first byte of the sequence header of the access unit referenced by the PTS_EP_start in the AV stream. Alternatively, if EP_type is equal to 1 ('audio'), this field indicates the relative address of the source packet containing the first byte in the audio stream of the access unit referenced by the PTS_EP_start in the AV stream.

RSPN_EP_start is of a size which is based on the source packet number as a unit, and is counted from the first source packet of the AV stream file, with the value of the offset_SPN, defined in ClipInfo(), as an initial value. The absolute address in the AV stream file is calculated by

SPN_xxx = RSPN_xxx – offset_SPN.

It is noted that the value of the RSPN_EP_start in the syntax must appear in the rising order.

The TU_map is now explained with reference to Fig.73. TU_map forms a time axis based on the source packet arrival time clock (timepiece of the arrive time base). This time axis is termed TU_map_time_axis. The point of origin of TU_map_time_axis is indicated by offset_time in the TU_map(). TU_map_time_axis is divided in a preset unit as from offset_time, this unit being termed time_unit.

In each time_unit in the AV stream, addresses on the AV stream file of the source packet in the first complete form are stored in TU_map. These addresses are termed RSPN_time_unit_start. The time at which begins the k(k ≥ 0)th time_unit on the TU_map_time_axis is termed TU_start_time(k). This value is calculated based on the following equation:

TU_start_time(k) = offset_time + k*time_unit_size.

It is noted that TU_start_time(k) has a precision of 45 kHz.

Fig.74 shows the syntax of TU_map. By way of explanation of the TU_map syntax shown in Fig.74, the 32-bit field of offset_time gives an offset time relative to TU_map_time_axis. This value indicates the offset time relative to the first time_unit in the Clip. The offset_time is of a size based on 45 kHz clock derived from the 27 MHz precision arrival time clocks as unit. If the AV stream is to be recorded as new Clip, offset_time must be set to 0.

The 32-bit field of time_unit_size affords the size of the time_unit, and is based

on 45 kHz clocks, derived from the 27 MHz precision arrival time clocks, as unit. Preferably, time_unit_size is not longer than one second (time_unit_size ≤ 45000). The 32 bit field of number_of_time_unit_entries indicates the number of entries stored in TU_map().

The 32-bit field of RSN_time_unit_start indicates the relative address of a site in the AV stream at which begins each time_unit. RSN_time_unit_start is of a size based on the source packet number as unit and is counted with the value of offset_SPN defined in ClipInfo() as from the first source packet of the AV stream file as an initial value. The absolute address in the AV stream file is calculated by

SPN_xxx = RSPN_xxx – offset_SPN.

It is noted that the value of RSN_time_unit_start in the for-loop of the syntax must appear in the rising order. If there is no source packet in the number (k+1) time_unit, the number (k+1) RSN_time_unit_start must be equal to the number k RSPN_time_unit_start.

By way of explanation of the ClipMark in the syntax of zzzzz.clip shown in Fig.45, the ClipMark is the mark information pertinent to clip and is stored in the ClipMark. This mark is not set by a user, but is set by a recorder (recording and/or reproducing apparatus 1).

Fig.75 shows the ClipMark syntax. By way of explanation of the ClipMark syntax shown in Fig.75, the version_number is four character letters indicating the version number of this ClipMark. The version_number must be encoded in accordance

with ISO 646 to "0045".

Length is a 32-bit unsigned integer indicating the number of bytes of the ClipMark() as from directly after the length field to the trailing end of ClipMark(). The number_of_Clip_marks is a 16-bit unsigned integer indicating the number of marks stored in ClipMark and may be equal to 0. Mark_type is an 8-bit field indicating the mark type and is encoded in accordance with the table shown in Fig.76.

Mark_time_stamp is a 32-bit field and stores the time stamp indicating a pointer having a specified mark. The semantics of mark_time_stamp differs with CPI_type in the PlayList(), as shown in Fig.77.

If CPI_type in CPI() indicates the EP_map type, this 8-bit field indicates the STC_sequence_id of the continuous STC domain where there is placed mark_time_stamp. If CPI_type in CPI() indicates TU_map type, this 8-bit field has no meaning but is set to 0. The 8-bit field of Character_set indicates the indicating method of character letters encoded in the mark_name field. The encoding method corresponds to the value shown in Fig.19.

The 8-bit field of name_length indicates the byte length of the mark name shown in the mark_name field. This mark_name field indicates the mark name. The byte number corresponding to the number of the name_length from left of this field is the effective character number and denotes the mark name. In the mark_name field, the values next following these effective character letters may be arbitrary.

The field of ref_thumbnail_index indicates the information of the thumbnail

picture appended to the mark. If the ref_thumbnail_index field is of a value different from 0xFFFF, a thumbnail picture is added to its mark, with the thumbnail picture being stored in the mark.thumb file. This picture is referenced using the value of ref_thumbnail_index in the mark.thumb file. If the ref_thumbnail_index field is of a value equal to 0xFFFF, a thumbnail picture is not appended to its mark.

MakerPrivateData has already been explained with reference to Fig.22 and hence is not explained here specifically.

Next, thumbnail_information is explained. A thumbnail picture is stored in a menu.thmb file or in a mark.thmb file. These files are of the same syntax structure and own a sole Thumbnail(). The menu.thmb file stores a picture representing respective PlatyLists. The totality of menu thumbnails are stored in the sole menu.thmb file.

The mark.thmb file stores a mark thumbnail picture, that is a picture representing a mark point. The totality of mark thumbnails corresponding to the totality of PlayLists and Clips are stored in the sole mark.thmb file. Since the thumbnails are frequently added or deleted, the operation of addition and partial deletion must be executable readily and speedily. For this reason, Thmbnail() has a block structure. Picture data is divided into plural portions each of which is stored in one tn_block. One picture data is stored in consecutive tn_blocks. In the string of tn_blocks, there may exist a tn_block not in use. The byte length of a sole thumbnail picture is variable.

Fig.78 shows the syntax of menu.thmb and mark.thmb and Fig.79 the syntax of Thumbnail in the syntax of menu.thmb and mark.thmb shown in Fig.78. By way of

explanation of the syntax of Thumbnail, shown in Fig.79, version_number is four character letters denoting the version number of this Thumbnail(). Version_number must be encoded to "0045" in accordance with ISO 646.

Length is a 32-bit unsigned integer indicating the number of bytes of MakerPrivateData() as from directly at back of the length field up to the trailing end of Thumbnail(). Tu_block_start_address is a 32-bit unsigned integer indicating the leading end byte address of the first tn_block, in terms of the relative number of bytes as from the leading end byte of Thumbnail() as a unit. The number of relative bytes is counted from 0. Number_of_thumbnails is a 16-bit unsigned integer which gives the number of entries of a thumbnail picture contained in Thumbnail().

Tu_block_size is a 16-bit unsigned integer which gives the size of one tn_block, in terms of 1024 bytes as a unit. If, for example, tn_block_size = 1, it indicates that the size of one tn_block is 1024 bytes. Number_of_tn_blocks is a 116-bit unsigned integer indicating the number of entries of tn_block in this Thumbnail(). Thumbnail_index is a 16-bit unsigned integer indicating the index number of the thumbnail picture represented by the thumbnail information for one for-loop beginning from the thumbnail_index field. The value 0xFFFF must not be used as Thumbnail_index. This Thumbnail_index is referenced by ref_thumbnail_index in UIAppInfoVolume(), UIAppInfoPlayList(), PlayListMark() and ClipMark().

Thumbnail_picture_format is an 8-bit unsigned integer representing the picture format of a thumbnail picture and assumes a value shown in Fig.80. In the table, DCF

and PNG are allowed only in menu.thumb. The mark thumbnail must assume the value of "0x00" (MPEG-2 Video 1-picture).

Picture_data_size is a 32-bit unsigned integer indicating the byte length of a thumbnail picture in terms of bytes as a unit. Start_tn_block_number is a 16-bit unsigned integer indicating the tn_block number of the tn_block where data of the thumbnail picture begins. The leading end of the thumbnail picture data must coincide with the leading end of the tn_block. The tn_block number begins from 0 and is relevant to the value of a variable k in the for-loop of tn_block.

X_picture_length is a 16-bit unsigned integer indicating the number of pixels in the horizontal direction of a frame of a thumbnail picture. Y_picture_length is a 16-bit unsigned integer indicating the number of pixels in the vertical direction of a frame of a thumbnail picture. Tn_block is an area in which to store a thumbnail picture. All tn_block in the Thumbnail() are of the same size (fixed length) and are of a size defined by tn_block_sixe.

Figs.81A and 81B schematically show how thumbnail picture data are stored in tn_block. If, as shown in Figs.81A and 81B, the thumbnail picture begins at the leading end of tn_block, and is of a size exceeding 1 tn_block, it is stored using the next following tn_block. By so doing, data with a variable length can be managed as fixed length data, so that the editing of deletion can be coped with by simpler processing.

An AV stream file is now explained. The AV stream file is stored in the "M2TS" directory (Fig.14). There are two types of the AV stream file, namely a Clip

A stream file and a Bridge-Clip AV stream file. Both AV streams must be of the structure of DVR MPEG-2 transport stream file as hereinafter defined.

First, the DVR MPEG2 transport stream is explained. The structure of the DVR MPEG-2 transport stream is shown in Fig.82. The AV stream file has the structure of a DVR MPEG 2 transport stream. The DVR MPEG 2 transport stream is made up of an integer number of Aligned units. The size of the aligned unit is 6144 bytes (2048*3 bytes). The Aligned unit begins from the first byte of the source packet. The source packet is 192 bytes long. One source packet is comprised of TP_extra_header and a transport packet. TP_extra_header is 4 bytes long, with the transport packet being 188 bytes long.

One Aligned unit is made up of 32 source packets. The last Aligned unit in the DVR MPEG 2 transport stream is also made up of 32 source packets. Therefore, the DVR MPEG 2 transport stream ends at a boundary of the Aligned unit. If the number of the transport packets of the input transport stream recorded on a disc is not a multiple of 32, a source packet having a null packet (transport packet of PID = 0x1FFFF) must be used as the last Aligned unit. The file system must not use excess information in the DVR MPEG 2 transport stream.

Fig.83 shows a recorder model of the DVR MPEG 2 transport stream. The recorder shown in Fig.83 is a conceptual model for prescribing the recording process. The DVR MPEG 2 transport stream obeys this model.

The input timing of the MPEG 2 transport stream is now explained. The input

MPEG 2 transport stream is a full transport stream or a partial transport stream. The input MPEG 2 transport stream must obey the ISO/IEC13818-1 or ISO/IEC 13818-9. The number i byte of the MPEG 2 transport stream is input simultaneously at time t(i) to T-STD (transport stream system target decoder provided for in SO/IEC13818-1) and to the source packetizer. Rpk is an instantaneous maximum value of the input rate of the transport packet.

A 27 MHz PLL 52 generates a frequency of 27 MHz clock. The 27 MHz clock frequency is locked at a value of the program clock reference (PCR) of the MPEG 2 transport stream. An arrival time clock counter 53 counts the pulses of the 27 MHz frequency. Arrival_time_clock(i) is a count value of the arrival time clock counter at time t(i).

A source packetizer 54 appends TP_extra_ header to the totality of the transport packets to create a source packet. Arrival_time_stamp indicates the time when the first byte of the transport packet reaches both the T-STD and the source packetizer. Arrival_time_stamp(k) is a sampled value of the Arrival_time_clock(k) as represented by the following equation:

arrival_time_stamp(k) = arrival_time_clock(k)% 230

where k denotes the first byte of the transport packet.

If the time separation between two neighboring transport packets is 230/2 7000000 sec (about 40 sec) or longer, the difference of the arrival_time_stamp of the two transport packets should be set to 230/2 7000000 sec. The recorder is provided

for such case.

A smoothing buffer 55 smoothes the bitrate of the input transport stream. The smoothing buffer must not overflow. Rmax is the output bitrate of the source packet from the smoothing buffer when the smoothing buffer is not null. If the smoothing buffer is null, the output bitrate of the smoothing buffer is 0.

Next, the parameters of the recorder model of the DVR MPEG 2 transport stream are explained. The value of Rmax is given by TS_recording_rate as defined in ClipInfo() associated with the AV stream file. This value may be calculated from the following equation:

Rmax = TS_recording_rate*192/188

where the value of TS_recording_rate is of a size in bytes/second.

If the input transport stream is an SESF transport stream, Rpk must be equal to TS_recording_rate as defined in ClipInfo() associated with the AV stream file. If the input transport stream is not an SESF transport stream, reference may be made to values defined e.g., in a descriptor of the MPEG 2 transport stream, such as maximum_bitrate_descriptor or partial_stream_descriptor for this value.

If the input transport stream is an SESF transport stream, the smoothing buffer size is 0. If the input transport stream is not an SESF transport stream, reference may be made to values defined in the descriptor of the MPEG 2 transport stream, such as, for example, the values defined in the smoothing_buffer_descriptor, short_smoothing _buffer_descriptor or in the partial_transport_stream_descriptor.

For the recorder and the player (reproducing apparatus), a sufficient size buffer needs to be provided. The default buffer size is 1536 bytes.

Next, a player model of the DVR MPEG 2 transport stream is explained. Fig.84 shows a player model of the DVR MPEG 2 transport stream. This is a conceptual model for prescribing the reproduction process. The DVR MPEG 2 transport stream obeys this model.

A 27 MHz X-tal 61 generates the frequency of 27 MHz. An error range of the 27MHx frequency must be ±30 ppm (2 7000000 ± 810 Hz). The arrival time clock counter 62 is a binary counter for counting the pulses of the frequency of 27 MHz. Arrival_time_clock(i) is a count value of the arrival time clock counter at time t(i).

In the smoothing buffer 64, Rmax is the input bitrate of the source packet to the smoothing buffer when the smoothing buffer is not full. If the smoothing buffer is full, the input bitrate to the smoothing buffer is 0.

By way of explaining the output timing of the MPEG 2 transport stream, if the arrival_time_stamp of the current source packet is equal to 30 bits on the LSB side of arrival_time_clock(i), the transport packet of the source packet is removed from the smoothing buffer. Rpk is an instantaneous maximum value of the transport packet rate. The overflow of the smoothing buffer is not allowed.

The parameters of the player model of the DVR MPEG 2 transport stream are the same as those of the recorder model of the DVR MPEG 2 transport stream described above.

Fig.85 shows the syntax of the source packet. Transport_packet() is an MPEG 2 transport stream provided in ISO/IEC 13818-1. The syntax of TP_Extra-header in the syntax of the source packet shown in Fig.85 is shown in Fig.86. By way of explaining the syntax of the TP_Extra-header, shown in Fig.86, copy_permission_indicator is an integer representing the copying limitation of the payload of the transport packet. The copying limitation may be copy free, no more copy, copy once or copying prohibited. Fig.87 shows the relation between the value of copy_permission_indicator and the mode it designates.

Copy_permission_indicator is appended to the totality of transport packets. If the input transport stream is recorded using the IEEE1394 digital interface, the value of copy_permission_indicator may be associated with the value of EMI (encryption mode indicator). If the input transport stream is recorded without employing the IEEE1394 digital interface, the value of copy_permission_indicator may be associated with the value of the CCI embedded in the transport packet. If an analog signal input is self-encoded, the value of copy_permission_indicator may be associated with the value of CGMS-A of the analog signal.

Arrival_time_stamp is an integer having a value as specified by arrival_time_stamp in the following equation:

arrival_time_stamp(k) = arrival_time_clock(k)%230.

By way of defining the ClipAV stream, the ClipAV stream must have a structure of the DVR MPEG 2 transport stream defined as described above.

Arrival_time_clock(i) must increase continuously in the Clip AV stream. If there exists a discontinuous point of the system time base (STC base) in the Clip AV stream, arrival_time_clock(i) in the Clip AV stream must increase continuously.

The maximum value of the different of the arrival_time_clock(i) between the beginning and the end of the Clip AV stream must be 26 hours. This limitation guarantees that, if there is no discontinuous point in the system time base (STC base) in the MPEG 2 transport stream, the PTS (presentation time stamp) of the same value never appears in the Clip AV stream. The MPEG 2 system standard provides that the PTS has a wraparound period of 233/90000 sec (about 26.5 hours).

By way of defining the Bridge-Clip AV stream, the Bridge-Clip AV stream must have a structure of the DVR MPEG 2 transport stream defined as described above. The Bridge-Clip AV stream must include a discontinuous point of one arrival time base. The transport stream ahead and at back of the discontinuous point of the arrival time base must obey the encoding limitations and the DVR-STD as later explained.

The present embodiment supports the video-audio seamless connection between PlayItems being edited. Seamless connection between PlayItems guarantees "continuous data supply" to the player/decoder and "seamless decoding processing". The "continuous data supply" is the capability of guaranteeing data supply to the decoder at a bitrate necessary to prevent buffer underflow. In order to enable data to be read out from the disc as data real-time properties are assured, data is to be stored in terms of a continuous block of a sufficiently large size as a unit.

The "seamless decoding processing" means the capability of a player in displaying audio video data recorded on the disc without producing pause or gap in the playback output of the decoder.

The AV stream, referenced by the seamless connected PlayItems, is explained. Whether or not the seamless display of a previous PlayItem and the current PlayItem is guaranteed may be verified from the connection_condition field defined in the current PlayItem. There are two methods for seamless connection of PlayItems, that is a method employing Bridge-Clip and a method not employing Bridge-Clip.

Fig.88 shows the relation between the previous PlayItem and the current PlayItem in case of employing Bridge-Clip. In Fig.88, the stream data, read out by the player, is shown shaded. In Fig.88, TS1 is made up of shaded stream data of the Clip1 (Clip AV stream) and shaded stream data previous to RSPN_arrival_time_discontinuity.

The shaded stream data of Clip1 of TS1 is stream data from an address of a stream required for decoding the presentation unit corresponding to IN_item of the previous PlayItem (shown as IN-time1 in Fig.88) up to the source packet referenced by RSPN_exit_from_previous_Clip. The shaded stream data prior to RSPN_arrival_time_discontinuity of Bridge-Clip contained in TS1 is stream data as from the first source packet of Bridge-Clip up to the source packet directly previous to the source packet referenced by RSPN_arrival_time_discontinuity.

In Fig.88, TS2 is made up of shaded stream data of Clip 2 (Clip AV stream) and

shaded stream data subsequent to RSPN_arrival_time_discontinuity of Bridge-Clip. The shaded stream data as from the RSPN_arrival_time_discontinuity of Bridge-Clip contained in TS2 stream data from the source packet referenced by RSPN_arrival_time_discontinuity to the last source packet of Bridge-Clip. The shaded stream data of Clip2 of TS2 is stream data from the source packet referenced by RSPN_enter_to_current_Clip to the address of the stream required for decoding the presentation unit corresponding to OUT_time of current PlayItem (shown by OUT_time2 in Fig.88).

Fig.89 shows the relation between the previous PlayItem and the current PlayItem in case of not employing Bridge-Clip. In this case, the stream data read out by the player is shown shaded. In Fig.89,TS1 is made up of shaded stream data of the Clip1 (Clip AV stream). The shaded stream data of Clip1 of TS1 is data beginning at an address of a stream necessary in decoding a presentation unit corresponding to IN_time of the previous PlayItem, shown at IN_time1 in Fig.89 as far as the last source packet of Clip1.

In Fig.89, TS2 is shaded stream data of Clip2 (Clip AV stream).

The shaded stream data of Clip2 of TS2 is stream data beginning at a first source packet of Clip2 as far as an address of the stream necessary for decoding the presentation unit corresponding to OUT_time of current PlayItem (shown at OUT_time2 in Fig.89).

In Figs.88 and 89, TS1 and T2 are continuous streams of the source packet.

Next, the stream provisions of TS1 and TS2 and the connection conditions therebetween are scrutinized. First, encoding limitations for seamless connection are scrutinized. By way of limitations on the encoding structure of a transport stream, the number of programs contained in TS1 and TS2 must be 1. The number of video streams contained in TS1 and TS2 must be 1. The number of audio streams contained in TS and TS2 must be 2 or less. The numbers of the audio streams contained in TS1 and TS2 must be equal to each other. It is also possible for elementary streams or private streams other than those depicted above to be contained in TS1 and/or TS2.

The limitations on the video bitstream are now explained. Fig.90 shows a typical seamless connection indicated by a picture display sequence. In order for a video stream to be demonstrated seamlessly in the vicinity of a junction point, unneeded pictures displayed at back of OUT_time1 (OUT_time of Clip1) and ahead of IN_time2 (IN_time of Clip2) must be removed by a process of re-encoding the partial stream of the Clip in the vicinity of the junction point.

Fig.91 shows an embodiment of realizing seamless connection using BridgeSequence. The video stream of Bridge-Clip previous to RSPN_arrival_time_discontinuity is comprised of an encoded video stream up to a picture corresponding to OUT_time1 of Clip1 of Fig.90. This video stream is connected to the video stream of previous Clip1 and is re-encoded to form an elementary stream conforming to the MPEG2 standard.

The video stream of Bridge-Clip subsequent to

RSPN_arrival_time_discontinuity is made up of an encoded video stream subsequent to a picture corresponding to IN_time2 of Clip2 of Fig.90. The decoding of this video stream can be started correctly for connecting the video stream to the next following Clip2 video stream. Re-encoding is made such that a sole continuous elementary stream conforming to MPEG 2 standard will be formed. For creating Bridge-Clip, several pictures in general need to be re-encoded, whilst other pictures can be copied from the original Clip.

Fig.92 shows an embodiment of realizing seamless connection without employing BridgeSequence in the embodiment shown in Fig.90. The Clip1 video stream is comprised of an encoded video stream as far as the picture corresponding to OUT_time1 of Fig.90 and is re-encoded so as to give an elementary stream conforming to the MPEG2 standard. In similar manner, the video stream of Clip2 is made up of encoded bitstreams subsequent to the picture associated with IN_time2 of Clip2 of Fig.90. These encoding bitstreams are already re-encoded to give a sole continuous elementary stream conforming to the MPEG2 standard.

By way of explaining encoding limitations of the video stream, the frame rates of the video streams of TS1 and TS2 must be equal to each other. The video stream of TS1 must be terminated at sequence_end_code. The video stream of TS2 must commence at Sequence header, GOP Header and with an I-picture. The video stream of TS2 must commence at a closed GOP.

The video presentation units defined in a bitstream (frame or field) must be

continuous with a junction point in-between. No gap of the fields or frames are allowed to exist at junction points. In case of encoding employing 3-2 pulldown, it may be necessary to rewrite "top_field_first" and "repeat_first_field" flags. Alternatively, local re-encoding may be made to prevent field gaps from being produced.

By way of explaining encoding limitations on the audio bitstream, the audio sampling frequency of TS1 and that of TS2 must be equal to each other. The audio encoding method of TS1 and that of TS2 (for example, MPEG1 layer 2, AC–3, SESF LPCM and AAC) must be equal to each other.

By way of explaining encoding limitations on MPEG-2 transport stream, the last audio frame of the audio stream of TS1 must contain audio samples having a display timing equal to the display end time of the last display picture of TS1. The first audio frame of the audio stream of TS2 must contain an audio sample having a display timing equal to the display start timing of the first display picture of TS2.

At a junction point, no gap may be allowed to exist in a sequence of the audio presentation units. As shown in Fig.93, there may be an overlap defined by the length of the audio presentation unit less than two audio frame domains. The first packet transmitting an elementary stream of TS2 must be a video packet. The transport stream at the junction point must obey the DVR-STD which will be explained subsequently.

By way of explaining limitations on the Clip and Bridge-Clip, no discontinuities in the arrival time base are allowed to exist in TS1 or in TS2.

The following limitations are applied only to the case of employing the Bridge-

Clip. The Bridge-Clip AV stream has a sole discontinuous point in the arrival time base only at a junction point of the last source packet of TS1 and the first source packet of TS2. The SPN_arrival_time_discontinuity defined in ClipInfo() represents an address of the discontinuous point, which must represent the address referencing the first source packet of TS2.

The source packet referenced by RSPN_exit_from_previous_Clip defined in BridgeSequenceInfo() may be any source packet in Clip1. It is unnecessary for this source packet to be a boundary of the Aligned unit. The source packet referenced by RSPN_enter_to_current_Clip defined in BridgeSequenceInfo() may be any source packet in Clip2. It is unnecessary for this source packet to be a boundary of the Aligned unit.

By way of explaining limitations on PlayItem, the OUT_time of the previous PlayItem (OUT_time 1 shown in Fig.89) must represent the display end time of the last video presentation unit of TS1. The IN_time of the current PlayTime (IN_time2 shown in Fig.88 and 89) must represent the display start time of the first presentation unit of TS2.

By way of explaining the limitations on the data allocation in case of employing Bridge-Clip by referring to Fig.94, the seamless connection must be made to guarantee continuous data supply by the file system. This must be realized by arranging the Bridge-Clip AV stream, connecting to Clip1 (Clip AV stream file) and Clip2 (Clip AV stream file), such as to satisfy data allocation prescriptions.

RSPN_exit_from_previous_Clip must be selected so that the stream portion of Clip1 (Clip AV stream file) previous to RSPN_exit_from_previous_Clip will be arranged in a continuous area not less than half fragment. The data length of the Bridge-Clip AV stream must be selected so that the data will be arranged in the continuous area not less than half fragment. RSPN_enter_to_current_Clip must be selected so that the stream portion of Clip2 (Clip AV stream file) subsequent to RSPN_enter_to_current_Clip will be arranged in a continuous area not less than half fragment.

By way of explaining data allocation limitations in case of seamless connection not employing Bridge-Clip, by referring to Fig.95, the seamless connection must be made so as to guarantee continuous data supply by the file system. This must be realized by arranging the last portion of the Clip1 (Clip AV stream file) and the first portion of the Clip2 (Clip AV stream file) so that the provisions on data allocation will be met.

The last stream portion of Clip1 (Clip AV stream file) must be arranged in a continuous area not less than one half fragment. The first stream portion of Clip2 (Clip AV stream file) must be arranged in a continuous area not less than one half fragment. Next, DVR-STD is explained. This DVR-STD is a conceptual model for modeling the decoding processing in the generation and verification of the DVR MPEG 2 transport stream. The DVR-STD is also a conceptual model for modeling the decoding processing in the generation and verification of the AV stream referenced by

two PlayItems seamlessly connected to each other as described above.

Fig.96 shows a DVR-STD model. The model shown in Fig.96 includes, as a constituent element, a DVR MPEG 2 transport stream player model. The notation of n, Tbn, Mbn, Ebn, Tbsys, Bsys, Rxn, Rbxn, Rxsys, Dn, Dsys, On and P9(k) is the same as that defined in T-STD of ISO/IEC 13818-1, wherein n is an index number of an elementary stream and TBn is a transport buffer of the elementary stream n.

MBn is a multiplexing buffer of the elementary stream n and exists only for the video stream. EBn is an elementary stream buffer of the elementary stream n and is present only for the video stream. TBsys is a main buffer in a system target decoder for the system information for a program being decoded. Rxn is a transmission rate with which data is removed from TBn. Rbxn is a transmission rate with which the PES packet payload is removed from MBn and is present only for a video stream.

Rxsys is a transmission rate with which data is removed from TBsys. Dn is a decoder of the elementary stream n. Dsys is a decoder pertinent to the system information of a program being decoded. On is a re-ordering buffer of the video stream n. Pn(k) is a number k presentation unit of the elementary stream.

The decoding process for DVR-STD is explained. During the time a sole DVR MPEG 2 transport stream is being reproduced, the timing of inputting the transport packet to TB1, TBn or TBsys is determined by arrival_time_stamp of the source packet. The prescriptions for the buffering operation of TB1, MB1, EB1, TBn Bn, TBsys and Bsys are the same as those of the T-STD provided for in ISO/IEC 13818-1,

while the prescriptions for the deciding and display operations are also the same as the T-STD provided for in ISO/IEC 13818-1.

The decoding process during the time the seamlessly connected PlayLists are being reproduced is now explained. Here, the reproduction of two AV streams referenced by the seamlessly connected PlayItems is explained. In the following explanation, the reproduction of TS1 and TS2, shown for example in Fig.88, is explained. TS1 and TS2 are a previous stream and a current stream, respectively.

Fig.97 shows a timing chart for inputting, decoding and display of transport packets when transferring from a given AV stream (TS1) to the next AV stream seamlessly connected thereto (TS2). During transfer from a preset AV stream (TS1) to the next AV stream seamlessly connected thereto (TS2), the time axis of the arrival time base of TS2 is not the same as the time axis of the arrival time base of TS1 (indicated by ATC1 in Fig.97).

Moreover, the time axis of the system time base of TS2 (indicated by ATC1 in Fig.97) is not the same as the time axis of the system time base of TS1 (indicated by STC1 in Fig.97). The video display is required to be continuous seamlessly, however, there may be overlap in the display tine of the presentation units.

The input timing to DVR-STD is explained. During the time until time T1, that is until the inputting of the last video packet to the TB1 of DVR-STD, the input timing to the buffers of TB1, TBn or TBsys of DVR-STD is determined by arrival_time_stamp of the arrival time base of TS1.

The remaining packets of TS1 must be input to buffers of TBn or to TBsys of DVR-STD at a bitrate of TS_recording_rate (TS1). The TS_recording_rate (TS1) is the value of TS_recording_rate defined in ClipInfo() corresponding to Clip1. The time the last byte of TS1 is input to the buffer is the time T2. So, during the time between time T1 and time T2, arrival_time_stamp of the source packet is discounted.

If N1 is the number of bytes of the transport packet of TS1 next following the last video packet of TS1, the time DT1 from time T1 until time T2 is the time necessary for N1 bytes to be input completely at a bitrate of TS_recording_rate (TS1), and is calculated in accordance with the following equation:

DT1 = T2 – T1 = N1/TS_recording_rate.

During the time from time T1 until time T2 (TS1), both the values of RXn and RXsys are changed to the value of TS-recording_rate (TS1). Except this rule, the buffering operation is the same as that of T-STD.

At time T2, the arrival time clock counter is reset to the value of arrival_time_stamp of the first source packet of TS2. The input timing to the buffer of TB1, TBn or TBsys of DVR-STD is determined by arrival_time_stamp of the source packet of TB2. Both RXn and RXsys are changed to values defined in T-STD.

By way of explaining additional audio buffering and system data buffering, the audio decoder and the system decoder need to have an additional buffering amount (data amount equivalent to one second) in addition to the buffer amount defined in T-STD in order to allow input data of a domain from time T1 to time T2.

By way of explaining the video presentation timing, the display on the video presentation unit must be continuous, that is devoid of gaps, through junction point. It is noted that STC1 is the time axis of the system time base of TS1 (indicated as STC1 in Fig.9), while STC2 is the time axis of the system time base of TS2 (shown at STC2 in Fig.97; correctly, STC2 begins at time the first PCR of TS2 has been input to the T-STD).

The offset between STC1 and STC2 is determined as follows: the PTS1end is the PTS on STC1 corresponding to the last video presentation unit of TS2. PTS2start is PTS on STC2 corresponding to the first video presentation unit of TS2 and Tpp is the display time period of the last video presentation unit of TS1, the offset STC_delta between two system time bases is calculated in accordance with the following equation: STC_delta = PTS1end + Tpp – PTS2start.

By way of explanation of the audio presentation timing, there may be overlap in the display timing of the audio presentation unit, with the overlap being less than 0 to 2 audio frames (see "audio overlap" shown in Fig.97). The indication as to which of audio samples is to be selected and re-synchronization of the display of the audio presentation unit to the corrected time base at back of the junction point are set on the player.

By way of explaining the system time clock of DVR-STD, the last audio presentation unit of TS1 is displayed at time T5. The system time clock may be overlapped between time T2 and time T5. During this time domain, the DVR-STD

switches the system time clocks between the value of the old time base (STC1) and the value of the new time base (STC2). The value of STC2 may be calculated in accordance with the following equation:

$$\text{STC2} = \text{STC1} - \text{STC\_delta}.$$

The buffering continuity is explained. STC11video_end is the value of STC on the system time base STC2 when the first byte of the first video packet reaches TB1 of DVR-STD. STC22video_start is the value of STC on the system time base STC2 when the first byte of the first video packet reaches TB1 of DVR-STD. STC21video_end is the value of STC11video_end calculated as the value on STC2 of the system time base STC2. STC21video_end is calculated in accordance with the following equation:

$$\text{STC21video\_end} = \text{STC11video\_end} - \text{STC\_delta}.$$

In order to obey DVR-STD, the following two conditions must be met: First, the arrival timing of the first video packet of TS2 at TB1 must satisfy the following inequality:

$$\text{STC22video\_start} > \text{STC21video\_end} + \Delta\text{T1}.$$

If it is necessary to re-encode and/or multiplex the partial stream of Clip1 and/or Clip2, in such a manner that the above inequality will be satisfied, this re-encoding or multiplexing is performed as appropriate.

Second, the inputting of the video packet from TS1 followed by the inputting of the video packet from TS2 on the time axis of the system time base, mapped from

STC1 and STC2 on the same time axis, must not overflow or underflow the video buffer.

If the above syntax, data structure and the rules are used as basis, the contents of data recorded on the recording medium or the reproduction information can be managed properly to enable the user to confirm the contents of data recorded on the recording medium at the time of reproduction or to reproduce desired data extremely readily.

Next, detailed description will be given on recording of an AV stream file in a case where time_controlled_flag in the syntax of ClipInfo shown in Fig.46 is set to be 1. In a case where time_controlled_flag is set to be 1, time lapse of the AV stream and the volume of the data bytes of the AV stream have the following relationship. That is , it is assured that the time lapse will be proportionate to the volume of the data bytes of the AV stream within a preset error range.

$$TS\_average\_rate*192/188 * (t-\alpha) <= AV\_file\_size(t) <= TS\_average\_rate*192/188 * (t+\alpha) \quad \text{... equation (1)}$$

The above equation is substantially the same as one shown in the description of time_controlled_flag of ClipInfo in Fig.46 although its form is slightly different.

In the equation, TS_average_rate is an average bit rate of the AV stream file (DVR transport stream file) shown by a unit of bytes/seconds, and is indicated by a field of the same name in the ClipInfo. Also, t is lapse time of the arrival time base

from the first source packet of the AV stream file shown by a unit of seconds. AV_file_size(t) is the size of the AV stream file at time t shown by a unit of bytes. α is a predetermined fixed value, and e.g., 300 seconds.

The value of TS_average_rate is arbitrarily determined depending on the application of the recorder. For example, the values of TS_average_rate are respectively determined for the respective modes in accordance with the recording modes such as a long-time picture recording mode (LP mode), a standard picture recording mode (SP mode), and a high quality picture recording mode (HQ mode).

In a case where the AV stream file is recorded to satisfy the equation (1), if the stream is partially erased for a certain time of the stream, it can be assured to generate on the disc an unoccupied area recordable for a time equal to the erased time at a bitrate indicated by the TS_average_rate of the stream in question. For example, if the stream is partially erased for a certain time of the AV stream in the SP mode, it is possible to generate on the disc an unoccupied area recordable for a time equal to the erased time in the same SP mode.

Fig.98 is a block diagram for illustrating the operation of an AV encoder 15 of the recording and/or reproducing apparatus 1 of Fig.1 in case the variable bit rate is controlled so that the time lapse will be proportionate to the volume of the data bytes of the AV stream within a preset error range. The blocks depicted with the same numerals in Fig.98 are the same as those in Fig.1.

First, recording modes, such as LP or SP modes, are input through a user

interface 24 to the controller 23. The controller 23 is responsive to the recording mode to set average bit rates of video encoding and multiplexing bitrate of the AV stream (DVR transport stream) to be recorded (step S20 of the flowchart of Fig.99).

The controller 23 sets time_controlled_flag to 1 to set the average bitrate and the multiplexing bitrate of the multiplexed stream to TS_average_rate and TS_recording_rate, respectively. The controller 23 outputs a database of the Clip information file in the ClipInfo of which are set time_controlled_flag, TS_recording_rate and TS_average_rate. The Clip Information file is recorded through the processing of the ECC encoding unit 20 explained in Fig.1.

In encoding an analog video input, video is input from terminal 11. In transcoding a video input of the digital broadcast, video from the AV decoder 27 is input. The input video is input to a video encoder 151. The controller 23 calculates the amount of encoding bits allocated to video per preset time to designate it for the video encoder. The video encoder 115 encodes video per preset time to input the amount of bits actually generated to the controller 23. The preset time size is the video GOP and is equal to 0.5 sec. Based on the cumulative value as from the start of encoding of the amount of actually generated encoding bits input from the encoder, the controller 23 controls the variable bitrate of video encoding, so that the time lapse of the AV stream will be proportionate to the data byte volume of the AV stream within the range of a preset error to calculate the amount of the encoded bits for the video for the next preset time. If the controller 23 can be furnished with the relative video encoding difficulty

(such as magnitude of prediction residues of motion vector prediction or the magnitude of the quantization scale of DCT coefficients) form the encoder, a variable bitrate can be realized to a higher picture quality. That is, control is made so that the higher the relative video encoding difficulty, the larger becomes the amount of allocated encoding bits to the video per preset time.

The video encoder 115 inputs a video stream to a multiplexer 16 which is also fed with an audio stream and with the system information (S), such as AV synchronization. The encoding processing flow for an audio input or the system information (S) such as AV synchronization is the same as the explanation of Fig.1.

The multiplexer 16 multiplexes video and audio streams to a transport stream of a preset multiplexing bitrate. In packetizing video and audio, control must be made so as not to disrupt the system target decoder (T– STD) of the MPEG–2 transport stream. If, due to T– STD limitations, it is not possible to packetize the video access unit (pictures of encoded I, P and B pictures) nor audio access units (audio frames), the multiplexer 16 multiplexes so as not to produce a null-packet (packet which is 0x1FFF). By this multiplexing control, the time interval between contiguous transport packets becomes irregular, such that packets are intermittently produced.

The transport packet, output from the multiplexer 16, is input to the source packetizer 19, which then appends an arrival time stamp to each transport packet to form a source packet. A source packet string is front-padded to generate an AV stream file, which then is recorded on the recording medium through the processing in the

ECC encoding unit 20.

Fig.99 is a flowchart for illustrating the operation of recording the AV stream by variable bit rate encoding of the video in the encoding mode of guaranteeing that the time lapse of the AV stream will be proportionate within a preset error range to the amount of the data bytes of the AV stream (time_controlled_flag = 1).

At step S20, the controller 23 sets the average bitrate of video encoding and multiplexing bitrate of the transport stream TS_recording_rate.

The average bitrate of video encoding is the TS_average_rate less the constant bitrate of audio encoding less multiplexing overhead bitrate. It is noted that TS_average_rate is set to a predetermined value depending on the recorder application (LP or SP mode).

TS_recording_rate is to be larger than the maximum bitrate of video encoding at a variable bitrate plus the constant bitrate for audio encoding plus the multiplexing overhead bitrate.

At step S21, the controller 23 controls the video encoder 151 so that a vide stream will be encoded at a variable bitrate in such a manner that a preset average bitrate will be guaranteed from one preset time domain to another.

At step S22, the controller 23 controls the multiplexer 16 so as not to produce a null-packet in case there is no elementary stream to be rendered into a transport packet. By this multiplexing control, the time interval between two contiguous transport packets becomes irregular, such that packets are intermittently produced.

At step S23, the controller 23 appends an arrival time stamp to each transport packet to form a source packet by way of controlling the source packetizer 19. A source packet string is front-padded to generate an AV stream file, which then is recorded.

The method of controlling the VBV (video buffering verifier) of MPEG in case of variable bitrate encoding of video. It is noted that VBV is a theoretical decoder model prescribed by MPEG (see Fig.100). It is up to the MPEG encoder to encode the video stream so that VBV will operate correctly. This limits the encoding method (mainly quantization control and limitations on the amount of bits for a picture). A buffer owned by VBV is termed a VBV buffer. This represents the theoretically necessary minimum buffer size for a real decoder. In the case of the MPEG2 main profile main level, the VBV buffer size is 1.75 Mbits.

As for VBV of MPEG for a variable bitrate, a method shown in Fig.101 is widely known. That is, Fig.101 illustrates VBV control in case the input bitrate to the buffer becomes the maximum bitrate if the variable bitrate in the VBV buffer is not full, with the input bitrate to the buffer becoming zero if the variable bitrate in the VBV buffer is full. In Fig.101, the tilt of a rightward ascending line represents the maximum bitrate of VBR. If the VBV buffer is not full, the buffer takeup amount is increased at the maximum bitrate of VBV. On the other hand, if the bit occupying amount in the VBV buffer is full, the input bitrate to the buffer is 0, with the buffer takeup amount remaining unchanged. The abscissa denotes the time axis, T1 indicating a decoding

time point. At time T1, a picture of the time point T1 is instantly decoded, with the buffer takeup amount being decreased. Subsequently, a picture is decoded each preset time interval to decrease the buffer takeup amount. In the method shown in Fig.101, there is no possibility of a video decoder generating stuffing bytes in the video stream.

On the other hand, according to the present invention, the VBV is controlled as shown in Fig.102. That is, in variable bitrate changing the bitrate every preset time interval, such as every GOP, VBV control is performed at CBR (constant bitrate) within a preset time. Fig.102 shows VBV control in case of CBR control within a GOP (such as 0.5 sec video sequence). That is, Fig.102 illustrates VBV control in case the the input bitrate to the VBV buffer is the encoding bitrate for the current GOP and in case a stuffing byte is inserted so as not to produce overflow of the VBV buffer.

The following sequence is used in verifying whether or not the stuffing byte is to be inserted and in calculating the amount of the stuffing bytes in case the stuffing byte is inserted. In the following explanation, it is assumed that

VBV_BUFFER_SIZE = 1.75*1024*1024 bits

gop_bit_rate: bitrate per GOP [bit/second].

(1) Calculations of the minimum amount of bits of a picture being encoded

Take a picture at time d1 of Fig.102 as an example. First, the bit occupying amount vbv_b of the VBV buffer immediately before decoding by the VBV of the picture at time d1 is obtained. Then, a sum tmp corresponding to the bit occupying amount vbv_d plus the amount of bits input at a bitrate gop_bit_rate as from time d1

until the decoding time d2 of the next picture (tau) is calculated. The minimum amount of bits of the picture currently encoded may be found from tmp and VBV_BUFFER_SIZE as follows:

tmp = vbv_b + gop_bit_rate*tau

min_picture_bit = tmp – VBV_BUFFER_SIZE.

(2) Check as to whether or not byte stuffing is necessary following picture encoding

If the actual encoding bit gen_picture_bit of the current picture is smaller than min_picture_bit, stuffing bytes of a size indicated by the following formula are produced. After the picture currently being encoded, the video encoder encodes a number of stuffing bytes equal to the number of num_stuffing_byte. One stuffing byte is an 8-bit code "0000 0000".

if (gen_picture_bit < min_picture_bit)

num_stuffing_byte = (min_picture_bit– gen_picture_bit + 4)/8.

In the method shown in Fig.102, it is targeted that control is made so that the video encoder uses the amount of bits allocated to the video of a preset time. The input bitrate to the VBV buffer is the encoding bitrate of the current GOP, with the video encoder then generating stuffing bytes so as not to produce overflow in the VBV buffer.

The VBV control is effective in order that the relation between the time lapse of the AV stream and the amount of bytes in the AV stream will be linear within a preset error range, as shown in Fig.103, according to the concept of the present

invention. If the VBV control shown in Fig.101 is used, the relation shown in Fig.103 cannot be guaranteed in the presence of a still picture continuing for long time in the input video. That is, the still picture has a smaller amount of the information, so that, if the amount of allocated bits for encoding is made larger than the amount of information, the amount of bits generated on actual encoding is saturated at a smaller value. Thus, in such case, the relation between time lapse of the AV stream and the amount of data bytes of the AV stream is not linear, as shown in Fig.104. If, in such case, VBV control shown in Fig.102 is used, the relation between time lapse of the AV stream and the amount of data bytes of the AV stream can positively be linear within a preset error range because the video encoder aimed to manage control to use the amount of bits allocated to video of a preset time generates stuffing bytes so that the input bitrate to the VBV buffer will be the encoding bitrate of the current GOP and so that no overflow will be produced in the VBV buffer.

In the case of Fig.104, if the AV stream of the time portion of the still picture is erased, it is not possible to produce an unoccupied area recordable for a time corresponding to the erased time portion with a bitrate indicated by the TS_average_rate of the stream on the disc, because the amount of data bytes of the time portion is smaller than the data size corresponding to the average bitrate multiplied with the erasure time. On the other hand, if the stream is partially erased for a certain time of the AV stream, it is possible to generate on the disc an unoccupied area recordable for a time equal to the erased time at a bitrate indicated by the

TS_average_rate of the stream in question.

Fig.105 shows a flowchart for illustrating the detailed processing of variable bitrate control of video in the processing at step S21 of Fig.99 described above.

At step S200, an initial value SV1 is set in an allowance value sv_now of VBR. According to the present invention, variable bitrate control is managed so that, for guaranteeing that the relation between the time lapse of the AV stream and the amount of data bytes of the AV stream will be linear within a preset error range, the allowance amount sv_now of VBR will be in a range between 0 and the maximum value SVMAX.

If, in the above equation (1), $\alpha = 300$ sec, SV1 and SVMAX are of the following values:

SV1 = (average bitrate of video)*300

SVMAX = SV1*2

where the average bitrate of video is of a value as determined at step S20 of Fig.99 (see Fig.107).

At step S201, the allocated amount of bits of encoding of the current GOP is calculated.

At step S202, it is checked whether or not the following inequality:

$sv\_now + b\_av - b\_alloc \leq 0$

holds. This check is performed to verify whether or not the allowance value of VBR is negative (–).

In the above inequality, b_av is an average value of the amount of allocated

encoding bits per GOP, as found from the average encoding bitrate of video. With the GOP time duration of 0.5 sec, b_av is of the following value:

b_av = (average bitrate of video)*0.5.

If the result of check at step S202 is YES, processing transfers to step S203. If the result of check at step S202 is NO, processing transfers to step S204, where b_alloc is set to b_av. Processing then transfers to step S205.

At step S203, it is checked whether or not the following inequality holds:

sv_now + b_av – b_alloc ≤ SVMAX

this step being a check as to whether or not the allowance value of VBR exceeds the maximum value SVMAX.

If the result of check at step S203 is YES, processing transfers to step S205. If the result of check at step S203 is NO, processing transfers to step S204 to set b_alloc to b_av. Then, processing transfers to step S205.

At step S205, the current GOP is encoded. Control is managed so that the current GOP is encoded with the amount of allocated bits b_alloc and so that no overflow will be produced in the VBV buffer. This processing is explained in detail with reference to Fig.106.

At step s206, the allowance value of VBR sv_now is updated in accordance with the following equation:

sv_now += b_av – b_gen

where b_gen is the amount of encoding bits of the current GOP obtained as a result of

encoding of the current GOP.

At step S207, it is checked whether or not the current GOP is the last GOP. If the result of check at step S207 is, processing comes to a close. If the result of check at step S207 is NO, processing reverts to step S201.

Fig.106 is a flowchart for illustrating detailed processing of VBV control in the processing at step S205 of Fig.105.

At step S300, the amount of encoding bits allocated to the current GOP is converted to the encoding bitrate gop_bit_rate in accordance with the following equation:

gop_bit_rate = b_alloc/(15/29.97).

At step S301, the minimum amount of bits of the picture being encoded in the current GOP min_picture_bit is found by the following equation:

tmp = vbv_b + gop_bit_rate*tau

min_picture_bit = tmp – VBV_BUFFER_SIZE.

In the above equation, vbv_b is the bit occupying amount of the VBV buffer directly previous to the decoding of the picture currently encoded by VBV (see Fig.102).

It in noted that tau is the difference between the decoding time of the picture currently encoded and the decoding time of the next picture (see Fig.102).

It is noted that VBV_BUFFER_SIZE is the VBV buffer size and, in the case of MPEG2 MP@ML, is 1.75 Mbits.

At step S302, the current picture is encoded to produce the amount of generated bits gen_picture_bit.

At step S303, the following inequality:

gen_picture_bit < min_picture_bit

is checked.

If the result of check at step S303 is YES, processing transfers to step S304. If the result of check at step S303 is NO, processing transfers to step S305.

At step S304, the video encoder encodes a number of stuffing bytes corresponding to the number of num_stuffing_byte after the currently encoded picture and appends the encoded stuffing bytes at back of an encoded picture (see Fig.102):

num_stuffing_byte = (min_picture_bit– gen_picture_bit+4)/8.

At step S305, it is checked whether or not the picture is the last picture of the GOP. If the result of check at step S305 is YES, processing comes to a close. If the result of check at step S305 is NO, processing reverts to step S301.

By controlling the variable bitrate encoding of the video stream as described above to generate an AV stream file, it can be guaranteed that the relation between time lapse of the AV stream and the amount of data bytes of the AV stream be linear within a preset error range. If in this manner a certain time portion of the stream is partially erased, it is possible to produce an unoccupied area, in an amount corresponding to the erased time portion, which unoccupied area is recordable with a bitrate indicated by TS_average_rate of the stream in question.

For comparison, two exemplary recording methods for an AV stream, in an encoding mode not guaranteeing the proportional relation between time lapse of the AV stream and the amount of data bytes of the AV stream (time_controlled_flag = 0), are now shown.

The first case of time_controlled_flag = 0 is a case of transparently recording a transport stream of the AV stream (program) of digital broadcast. If digital broadcast uses statistic multiplexing, an AV stream therein is usually of a variable bitrate. The relation between time lapse of the AV stream and the amount of data bytes of the AV stream in general is not necessarily linear, so that, if Clip is prepared on transparent recording of this AV stream, time_controlled_flag is set to 0.

The second case of time_controlled_flag = 0 is a case of encoding the video at a variable bitrate so that the bitrate will be not higher than a preset average bitrate. This is a case in which VBV control of video encoding is performed, such that, if the VBV buffer is not full or full, the input bitrate to the buffer will be the maximum bitrate or zero, respectively. The AV stream recording method in this case is now explained with reference to Figs.108 and 109.

Fig.108 shows a flowchart for explaining the operation of encoding the video at a variable bitrate to record the AV stream in an encoding mode not guaranteeing the proportional relation between time lapse of the AV stream and the amount of data bytes of the AV stream.

The flowchart is the same as that of Fig.99 except step S400.

At step S400, the video encoder 151 is controlled so that the video stream will be encoded at a variable bitrate in such a manner that the bitrate will be lower than a preset average bitrate from one preset time domain to another.

Fig.109 is a flowchart for illustrating detailed processing of variable bitrate control of video in the processing of step S400 of Fig.108.

At step S500, an initial value SV1 is set in the allowance value sv_now of VBR. The variable bitrate control in this case is made so that the allowance value sv_now will not be negative (–).

At step S501, the allocated bit b_alloc of encoding of the current GOP is found.

At step S502, it is checked whether the following inequality:

$sv\_now + b\_av - b\_alloc \geq 0$.

holds. This step S502 is a check step for verifying whether or not the allowance value of VBV is not minus.

In the above inequality, b_av is an average value of the amount of encoded bits of encoding per GOP. If the time duration of a GOP in question is 0.5 sec, b_av has the following value:

b_av = (average bitrate of video)*0.5

If the result of check at step S502 is YES, processing transfers to step S504.. If the result of check at step S502 is NO, processing transfers to step S504 to set b_alloc to b_av. Then, processing transfers to step S504.

At step s504, the current GOP is encoded. The current GOP is encoded with the

amount of allocated bits b-alloc. The VBV control in this case is such that, if the bit occupying value in the VBV buffer is not full or full, the input bitrate to the buffer will be the maximum bitrate of VBR (variable bitrate) or zero, respectively (see Fig.101). At this step, the stuffing bytes are not encoded in the video stream.

At step S505, the allowance value of VBR sv_now is updated in accordance with the following formula:

sv_now += b_av – b_gen

where b_gen is an amount of encoding bits of the current GOP obtained as a result of encoding the current GOP at step S504.

At step S506, it is checked whether or not the current GOP is the last GOP. If the result of check at step S506 is YES, processing comes to a close. If the result of check at step S506 is NO, processing reverts to step S501.

The recording method of Figs.108, 109 fails to guarantee the proportional relation between time lapse of the AV stream and the amount of data bytes of the AV stream within a preset error range. For example, if there is a still picture continuing for long time in the input video, the relation between time lapse of the AV stream and the amount of data bytes of the AV stream is as shown in Fig.104. That is, since the still picture is of a smaller information volume, the amount of bits generated on actual encoding is saturated at a smaller value, even if the amount of allocated bits for encoding is larger than the information volume. So, the relation between time lapse of the AV stream and the amount of data bytes of the AV stream in ths case is not linear.

On the other hand, if, with a view to managing control so that the video encoder will use the amount of bits allocated of preset time, control is exercised so that the input bitrate to the VBV buffer is the encoding bitrate for the current GOP, and so that the stuffing bytes will be generated by the video encoder so as not to produce VBV buffer overflow, it can be guaranteed that the relation between time lapse of the AV stream and the amount of data bytes of the AV stream is substantially linear within a preset error range.

As a method of realizing the encoding mode guaranteeing the proportional relation between time lapse of the AV stream and the amount of data bytes of the AV stream, it may be contemplated to insert a null-packet in multiplexing a transport packet to record a transport stream of a constant bitrate. This is an encoding method mainly used in recording a transport packet, such as D-VHS. Meanwhile, a null-packet means a transport packet the packet ID (PID) of which is set to 0x1FFF and which has no meaning as information.

For reference in comparing the present method to the method of Fig.99, Fig.110 shows a flowchart of the encoding mode in which, by encoding the transport stream of a preset bitrate, the proportional relation between time lapse of the AV stream and the amount of data bytes of the AV stream is guaranteed.

At step S600, the multiplexing bitrate of the transport stream and the bitrate for video encoding are set. At step S601, the video stream is encoded at a preset constant bitrate or at a bitrate lower than the present bitrate.

At step S602, a null-packet (a transport packet having no meaning as information) is generated and multiplexed, in case there lacks an elementary stream to be rendered into a transport packet, to encode a transport stream of a preset constant multiplexing bitrate.

At step S603, an arrival time stamp is appended to each transport packet to form a source packet, which source packet is then recorded on a recording medium.

If the AV stream is recorded as Clip by the above-described recording method, time_controlled_flag of the Clip is set to 1. However, this method, employing a null-packet and which thus fails to use the encoding bits for video encoding efficiently, suffers from the problem of the picture quality being inferior to that achieved with the encoding method of Fig.99 (this has been discussed in detail in e.g., the description of the prior art of Japanese Laying-Open Patent Publication H-11-220727). So, the recording method of Fig.111 is not recommended in the present invention.

The method of partially deleting only a certain time portion of the AV stream file is hereinafter explained.

Fig.111 shows examples of an original AV stream file and an AV stream file following editing for erasing a partial reproducing portion of the AV stream file. It is assumed that, prior to the editing, Virtual PlayList points to IN_time and OUT_time on the original AV stream file. If an editing of erasing the stream portion not used by the Virtual PlayList (minimizing editing) is performed, it changes the original AV stream file into an as-edited stream file shown in Fig.111. The data as from the leading

end up to point X of the original AV stream file and the data as from a point Y to the trailing end are erased. In the following, a typical method of determining these points X and Y is explained.

Fig.112 illustrates the method of erasing unneeded data ahead of the point IN without analyzing the AV stream contents. PlayList denotes the point IN on the original AV stream file, while showing EP_map of the AV stream. For decoding a picture denoted by the point IN, an I-picture beginning from an address ISA2 is required.

At back of the point X, PAT, PMT and PCR packets are required. The PTS of RSPN_EP_start = ISA1 is pts1, while the PTS of RSPN_EP_start = ISA2 is pts2. If the time difference of the system time base of pts1 and that of pts2 is not less than 100 msec, there exist PAT, PMT and PCR packets between the address ISA1 and ISA2 (this holds at least for SESF, DVS,, ATSC and ISDB).

Therefore, the point X is determined ahead of the address ISA1. The point X must be on a boundary of an aligned unit. The recorder is also able to determine the point X, using EP_map, without analyzing the AV stream contents, by the following steps:

(S1) the step of finding SPN_EP_start having a value of PTS of the past display time closest to PTS of IN time on the system timebase; and

(S2) the step of finding SPN_EP_start having a value of PTS of the display time more past at least 100 msec than the value of the PTS of SPN_EP_start as found at step S1;

(S3) the point X being determined before SPN_EP_start as found at step S2, with the point X necessarily lying on the boundary of the aligned unit.

This method is simple because it is unnecessary to read out data of the AV stream to analyze its contents. However, there are occasions where unneeded data is left in reproducing the PlayList of the as-edited AV stream. If the AV stream data are read out in order to determine the point X and the data contents are analyzed, data unneeded for reproducing the PlayList can be erased efficiently.

Fig.113 illustrates the method of erasing unneeded data at back of the OUT point without analyzing the AV stream contents. PlayList points to the OUT point on the original AV stream, while showing EP_map of the AV stream.

It is presupposed that the video sequence beginning from SPN_EP_start = ISA4 is

I2 B0 B1 P5 ···

where I, P and B denote I-, P- and B-pictures, respectively and the suffix numbers denote the display order. If, in this processing, the recorder fails to analyze the AV stream contents, it is not aware of the information, such as picture coding type or temporal reference, of the pictures referenced by the PTS of OUT_time. The PTS of OUT_time may be referencing pictures B0 or B1(this cannot be known unless the recorder analyzes the AV stream contents). In such case, I2 is necessary in order to decode pictures B0 and B1. The PTS of I2 is larger than PTS of OUT time. The PTS of I2 is larger than PTS of OUT time (OUT_time < pts4, where pts4 is PTS of I2).

Although PTS of I2 is larger than PTS of OUT_time, I2 is required for B0 and B1.

So, the point Y is determined to be at back of an address ISA5 shown. ISA5 is the value of SPN_EP_start directly at back of ISA4 in EP_map. The point Y also must lie on the boundary of the aligned unit.

The recorder may determine the point Y at the next step, using EP_map, without analyzing the AV stream contents, by the following steps:.

(S1) the step of finding SPN_EP_start having a value of PTS of the display time closest to and more future than PTS pf OUT time on the system time base; and

(S2) the step of finding SPN_EP_start lying directly at back of SPN_EP_start as found on the system time base;

(S3) the point Y being set so as to be more rearwards than SPN_EP_start as found at step S2. The point Y must be on the boundary of the aligned unit.

This method is simple because it is unnecessary to read out AV stream data to analyze its contents to determine the point Y. However, there are occasions where unneeded data is left in reproducing the PlayList of the as-edited AV stream. If the AV stream data are read out in order to determine the point Y, and the data contents are analyzed, data unneeded for reproducing the PlayList can be erased more efficiently.

Referring to the flowchart of Fig.114, a typical operation of the PlayList is explained. This processing is performed on the multiplexing stream analysis unit 18 of the recording and/or reproducing apparatus shown in Fig.1.

At step S11, the stream analysis unit 18 sets the PID of video of the AV program

recorded. If plural videos are included in the transport stream, respective video PIDs are set.

At step S12, the stream analysis unit 18 receives a video transport packet.

At step S13, the stream analysis unit analyzes whether or not the payload of the transport packet (data part next following a packet header) begins with the first byte of the PES packet (PES packet is a packet prescribed in MPEG2 and packetize an elementary stream). This can be known by checking the value of "payload_unit_start_indicator" in the transport packet header. If this value is 1, the payload of the transport packet begins with the first byte of the PES packet. If the result of check at step S13 is NO, processing reverts to step S12. If the result of check at step S13 is YES, processing transfers to step S14.

At step S14, the stream analysis unit checks whether or not the payload of the PES packet begins at the first byte of sequence_header_code of MPEG video ("0x000001B3 in 32 bits"). If the result of check at step S14 is NO, processing reverts to step S12. If the result of check at step S14 is YES, processing transfers to step S15.

At step S15, the current transport packet is an entry point. At step S16, the stream analysis unit acquires the PID of the video to which belong the PTS of the I-picture beginning from the sequence_header_code and its entry point as well as the packet number of the above packet to input the video PID to the controller 23. The controller 23 forms EP_map.

At step S17, it is checked whether or not the current packet is the last input

transport packet. If the current packet is not the last packet, processing reverts to step S12. If the current packet is the last packet, processing is terminated.

The above-described sequence of operations may be executed not only on a hardware but also on a software. If the sequence of operations is to be carried out on the software, the program forming the software or a variety of programs are installed on a dedicated hardware of a computer, such that the programs are installed form a recording medium, such as a general-purpose personal computer.

Referring to Fig.115, this recording medium may be constituted by a package medium distributed for furnishing the user with a program, in addition to a computer. The package medium may be exemplified by a magnetic disc 221 (inclusive of floppy disc), an optical disc 222 (inclusive of CD-ROM (Compact Disc– Read Only memory) and DVD (Digital versatile Disc)), a magneto-optical disc 223 (inclusive of MD (Mini-Disc)) or a semiconductor memory 224. In addition, the recording medium may be constituted by a hard disc furnished to the user as it is pre-loaded on a computer and which includes a ROM 202 or a memory 208 having the program stored therein.

In the present specification, the respective steps stating the sequence of the program furnished by a medium includes not only the processing executed chronologically in accordance with the stated order but also the processing executed in parallel or batch-wise.

The system in the present specification denotes an entire apparatus made up of plural devices.

Industrial Applicability

In encoding and recording the AV stream, time_controlled_flag and TS_average_rate are recorded as the attribute information for the AV stream. If time_controlled_flag is set to 1, it is guaranteed that the relation between time lapse of the AV stream and the amount of data bytes of the AV stream is linear within a preset error range. TS_average_rate expresses the average bitrate of the AV stream file (transport stream) in terms of bytes/second as a unit, and is determined at a preset value by the application of the recorder. For example, TS_average_rate values are set for respective modes, that is long playing (LP) mode, standard playing (SP) mode, or high quality (HQ) mode.

If time_controlled_flag of the AV stream file is set to 1, and the stream is erased partially for a preset time stream portion, the unoccupied area corresponding to the time portion of the AV stream file of the SP mode, for example, can be created on the disc.

If time_controlled_flag is set to 1, the AV stream is encoded as follows:

(1) A multiplexing bitrate of a transport stream and an average bitrate for video encoding are set.

(2) A video stream is encoded at a variable bitrate such as to guarantee a preset average bitrate every preset time domain. It should be noted that the VBV (video buffering verifier) of MPEG video encoding is controlled in such a manner that, for managing control so that the video encoder will use a bit amount allocated to the video of the preset time, the input bitrate to the VBV buffer will be the current encoding bitrate, and

so that the video encoder will generate stuffing bytes to inhibit overflow of the VBV buffer.

(3) The multiplexing is controlled so that, in the absence of an elementary stream to be rendered into a transport packet, no null-packet will be produced.

(4) Arrival time stamp is appended to each transport packet to render it into a source packet and the source packet string is front-padded for recording as an AV stream file.

By encoding and recording the AV stream file in this manner, it can be guaranteed that, if a certain time portion of the stream is partially erased, an unoccupied area corresponding to the erased time portion and which is recordable at a bitrate shown by TS_average_rate can be created on the disc.